19-6-2012

KARRERA AMAIERAKO LANA – TRABAJO FIN DE MASTER

Trabajo Fin de Master: Iniciación a la Programación Visual e Interactiva y Desarrollo de Robótica Educativa con Scratch y Enchanting



Masterra Bigarren Hezkuntzako Irakasleen Prestakuntzan

Master para la Formación del Profesorado de Educación Secundaria

Tutorea -Tutor: Alfredo Pina

Ikaslea - Alumna: Ainhoa Yera Gil

ÍNDICE

INTRODUCCIÓN

El propósito de este trabajo es el de dar a conocer nuevas herramientas TIC para el desarrollo de contenidos de robótica, electrónica e informática en el área de Tecnología de 4º de E.S.O.

En el trabajo primeramente se describen las dificultades y carencias que afrontan los centros educativos a la hora de desarrollar estos contenidos a modo de justificación para el uso de nuevas herramientas.

A continuación se resumen los objetivos principales del trabajo.

Seguidamente, se realiza un análisis de los programas *Scratch* y *Enchanting,* dando cuenta de sus posibilidades y ventajas en lo referente a la implementación de contenidos relacionados con el área de Tecnología y la Informática.

Por último se realiza una propuesta de implementación del bloque de contenidos de Control y Robótica con *Scratch* y *Enchanting,* en la que se describen una serie de sesiones que incluyen ejemplos de programación, poniendo de manifiesto la dimensión práctica de las herramientas.

El trabajo pretende poner estas herramientas al alcance de los profesionales del ámbito educativo, ampliando el abanico de posibilidades a la hora de desarrollar contenidos didácticos del área tecnológica. El carácter gratuito de las mismas, junto con su compatibilidad con otras herramientas, las hace muy atractivas para su empleo en las aulas, más aún en el contexto actual de crisis económica. La secuenciación propuesta trata de facilitar al profesorado el acercamiento a las herramientas, a través de ejemplos de programación que simplifican la comprensión de su funcionamiento.

I. JUSTIFICACIÓN

En la actualidad, la dotación tecnológica de los centros educativos es una asignatura pendiente que los gobiernos no acaban de afrontar como una prioridad en sus partidas presupuestarias. Si bien es cierto que en los últimos años nuestras escuelas han experimentado una revolución tecnológica, sobre todo en la educación primaria con la incorporación de pizarras digitales, aún estamos lejos de alcanzar los niveles tecnológico-educativos del resto de países de la eurozona. El siguiente gráfico muestra el gasto público total en educación de los países europeos:

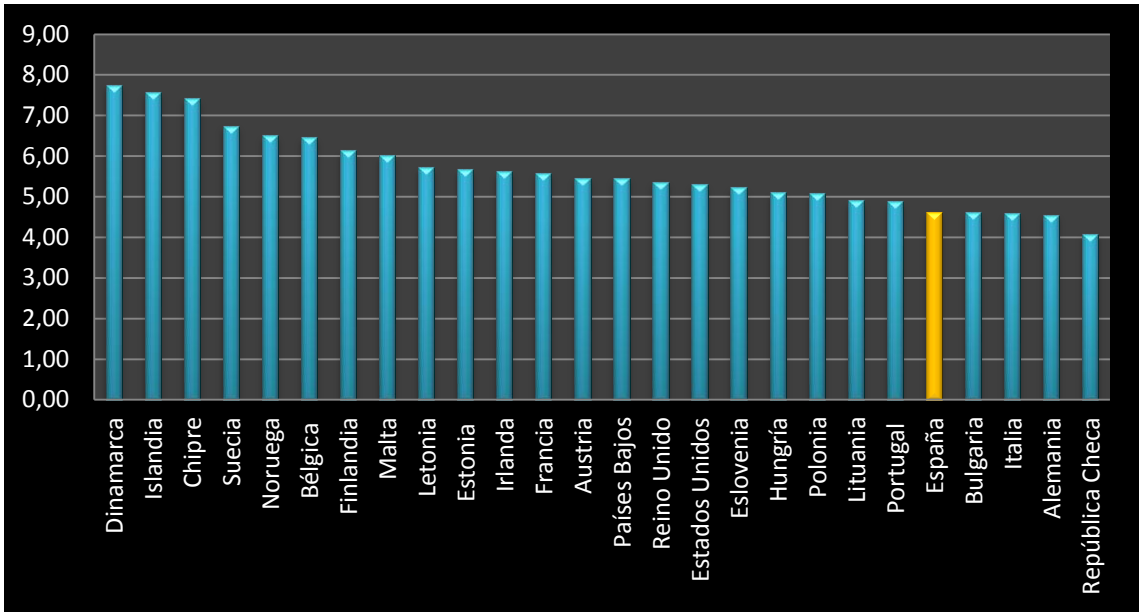


Gráfico 1: Gasto público en educación en relación al PIB en los países de la Unión Europea. Año 2008.

Como se puede ver, España se encuentra a la cola de Europa en gasto público para educación, solo por delante de la República Checa, Alemania, Italia y Bulgaria. En el caso de germano hay que tener en cuenta que según el FMI el PIB alemán per cápita en 2011 era un 30% más elevado que el de España.

En definitiva, la escasa financiación educativa en el Estado Español impide el abastecimiento tecnológico de los centros, así como la formación especializada del profesorado. Por ello, es importante fomentar el uso de herramientas TIC competitivas y a la vez baratas o gratuitas en educación, como la que se exponen en este trabajo. En el área informática, el desarrollo de software libre y de calidad en las últimas décadas ha sido espectacular y la comunidad educativa ha sabido aprovechar estos recursos. Sin embargo, el tipo de aplicaciones de software libre que más se emplean en educación, está más ligado a la gestión y difusión de contenidos (webct, blogs, aulario virtual etc.). Las herramientas que aquí se proponen, son de tipo software libre, pero cuentan con la ventaja de estar más orientadas hacia el aprendizaje específico de los contenidos didácticos.

Las herramientas que aquí se proponen, *Scratch* y *Enchanting*, se caracterizan por su gratuidad o su inferior coste en relación a otras herramientas y su versatilidad. *Enchanting* está orientado a la programación de robots y *Scratch* combinado con otras herramientas, puede usarse para la creación de contenidos multimedia interactivos o el control y simulación de placas electrónicas. Es decir, en conjunto son dos herramientas muy útiles para el desarrollo de contenidos propios de las áreas de Tecnología e Informática como son la robótica, la electrónica y la creación multimedia. Las siguientes tablas (*Tabla 1* y *Tabla 2*) destacan los

contenidos didácticos del currículum de 4º de E.S.O, correspondientes a las áreas de Tecnología e Informática, que se pueden desarrollar a través de las herramientas citadas:

AREA: TECNOLOGÍA	
BLOQUE DE CONTENIDOS	**CONTENIDOS DIDÁCTICOS**
CONTROL Y ROBÓTICA	**Experimentación con sistemas automáticos, sensores, actuadores y aplicación de la realimentación en dispositivos de control.**
	Diseño y construcción de robots.
	Uso del ordenador como elemento de programación y control. Trabajo con simuladores informáticos para verificar y comprobar el funcionamiento de los sistemas diseñados
ELECTRÓNICA	**Electrónica analógica. Componentes básicos, simbología, análisis y montaje de circuitos elementales.**
	Electrónica digital. Aplicación del álgebra de Boole a problemas tecnológicos básicos. Puertas lógicas.
	Uso de simuladores para analizar el comportamiento de los circuitos electrónicos.

Tabla1: Bloques de contenidos del área de Tecnología de 4º de E.S.O que podrían desarrollarse utilizando Scratch y Enchanting en combinación con otras herramientas.

ÁREA: INFORMÁTICA	
BLOQUE DE CONTENIDOS	**CONTENIDOS DIDÁCTICOS**
MULTIMEDIA	**Adquisición de imagen fija mediante periféricos de entrada.**
	Tratamiento básico de la imagen digital: los formatos básicos y su aplicación, modificación de tamaño de las imágenes y selección de fragmentos, creación de dibujos sencillos, alteración de los parámetros de las fotografías digitales: saturación, luminosidad y brillo.
	Captura de sonido y vídeo a partir de diferentes fuentes. Edición y montaje de audio y vídeo para la creación de contenidos multimedia.
	Las redes de intercambio como fuente de recursos multimedia. Necesidad de respetar los derechos que amparan las producciones ajenas

Tabla 2: Bloque de contenidos del área de Informática de 4º de E.S.O que podría desarrollarse utilizando Scratch.

Como se desprende de la *Tabla 2* el bloque de contenidos Multimedia excluye el aprendizaje de lenguajes de programación propicios para la creación multimedia. Por este motivo, en el presente trabajo se propone la utilización de *Scratch* que a través de su sencillo lenguaje de programación permite entre otras cosas crear elementos visuales e interactivos.

II. OBJETIVOS

El presente trabajo se propone los siguientes objetivos:

- Analizar las herramientas informáticas *Scratch* y *Enchanting* para el desarrollo de algunos bloques de contenidos de las áreas de Tecnología e Informática de 4º de E.S.O.
- Exponer ejemplos prácticos para la comprensión del funcionamiento de *Scratch* y *Enchanting*.
- Facilitar recursos didácticos para el desarrollo contenidos educativos relacionados con Control y Robótica, Electrónica o Multimedia que empleen software libre y sean de utilidad para los centros educativos con dificultades económicas.
- Realizar una propuesta de secuenciación para la implementación del bloque de contenidos de Control y Robótica haciendo uso de *Enchanting*.
- Aprovechar las similitudes entre *Scratch* y *Enchanting* para facilitar el aprendizaje intuitivo de lenguajes de programación orientados a la creación multimedia y a la programación robótica.

III. ANÁLISIS DE LAS HERRAMIENTAS TIC

En este capítulo se analizan las herramientas informáticas *Scratch* y *Enchanting*, describiendo entre otros aspectos, la instalación y las características, el nivel de compatibilidad con otras herramientas TIC usadas en educación, las posibilidades de implementación para el desarrollo de contenidos didácticos y el funcionamiento básico a través de ejemplos de programación.

III.I. SCTRATCH

III.I.I. INTRODUCCIÓN

Scratch es un entorno gráfico de programación desarrollado por un grupo de investigadores del *Lifelong Kindergarten Group* del Laboratorio de Medios del MIT, bajo la dirección del Dr. Mitchel Resnick.

El lenguaje de programación de Scratch se basa en las siguientes herramientas:

- **Micromundo de Logo:** Se utiliza para crear simulaciones y presentaciones multimedia, caracterizándose por su facilidad de operación, extensibilidad, interactividad, flexibilidad y seguimiento. Fue creado en 1967 por Seymour Papert, quien basándose el paradigma constructivista, lo concibió como una herramienta educativa para niños, siendo su personaje central una Tortuga.
- **E-toys de Squeak**: Se trata de un modelador de objetos que pueden ser ejecutados en varias plataformas. Incluye gráficos 2D y 3D, es posible insertar texto, presentaciones, videos, sonidos, MIDI, páginas web e imágenes entre otros. Al igual que Scratch, el modo de trabajo consiste en arrastrar y soltar bloques de instrucciones, en lugar de escribirlas.
- **LogoBlocks, CricketBlocks o PicoBlocks:**

  LogoBlocks es un lenguaje de programación gráfica desarrollado por el Grupo de Epistemología y Aprendizaje del Laboratorio de Medios del MIT para el ladrillo programable. El ladrillo programable es una pequeña computadora portátil que puede articularse con una construcción de LEGO para controlar motores y leer información capturada por sensores.

  PicoCricket: es un microprocesador que hace que mediante módulos añadidos, los objetos puedan girar, iluminarse y emitir sonidos así que es perfecto para que los niños desarrollen su creatividad con ayuda de la tecnología.

  PicoBlocks es un lenguaje de programación que funciona apuntando y haciendo clic, cortando y pegando. PicoBlocks parece un rompecabezas de piezas de colores que se ordenan y combinan en una pantalla de PC o Mac con un ratón. Alineando piezas etiquetadas en secuencias interconectadas se pueden crear comandos sencillos y complejos. Un "puntero" con cable USB enchufado al ordenador transmite los comandos a PicoCricket mediante una serie de destellos. Los motores y los sensores están conectados a PicoCricket, que a su vez funciona según la programación almacenada en su memoria de trabajo.

  Al igual que Scratch, estas herramientas utilizan bloques autoencajables de instrucciones, que se ajustan si son sintácticamente correctos, permitiendo al usuario centrar su atención en los algoritmos lógicos de programación.

Scratch consigue a través de una Interfaz cuidada y evolucionada, que iniciarse en el aprendizaje de la programación resulte más sencillo e intuitivo. Según sus creadores, fue diseñado como medio de expresión para ayudar a niños y jóvenes a expresar sus ideas de

forma creativa, al tiempo que desarrollan habilidades tanto de pensamiento lógico, como de aprendizaje para el Siglo XXI.

Scratch permite crear fácilmente historias interactivas propias, animaciones, juegos, grabar sonidos y realizar creaciones artísticas. En definitiva, la herramienta permite aprender a programar un ordenador de manera divertida.

Cuando se trabaja con Scratch se comprenden fácilmente conceptos matemáticos e informáticos que están muy bien integrados en el programa, como son:

- Los procesos interactivos (bucles),
- Los criterios condicionales (si, entonces, si-no),
- Las coordenadas en un plano,
- Las variables, etc.

Además el aprendizaje de desarrolla en contexto significativo y motivador ya que las variables se utilizan para el control de la visualización de una animación o en juego que uno mismo está construyendo.

Para instalar Scratch podemos acceder al enlace de descarga disponible en la página oficial de Scratch: http://scratch.mit.edu/ . Para conocer más acerca de la instalación consultar el ANEXO 1: INSTALACIÓN DE SCRATCH

III.I.II. INTERFAZ

Scratch dispone de una interfaz intuitiva, bien distribuida y de diseño animado. La imagen inferior (*Imagen 1*) muestra las distintas zonas que se distinguen en la Interfaz:

Imagen 1: Zonas en las que se divide la Interfaz de Scratch.

La siguiente tabla (*Tabla 3*) resume las opciones disponibles en cada una de las zonas:

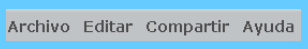
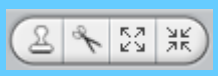
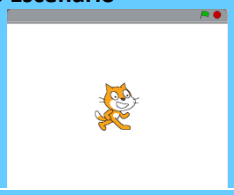
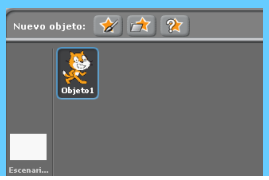
Zona de la Interfaz	Opciones	
1.1 Opciones de acceso rápido 	Idioma	
	Guardar	
	Publicar	
1.2 Menús Principales 	Archivo	
	Editar	
	Compartir	
	Ayuda	
1.3 Herramientas básicas para Objetos 	Duplicar	
	Agrandar	
	Achicar	
1.4 Vista del Escenario 	Escenario Pequeño	
	Escenario Completo	
	Modo Presentación	
2.1 Paleta de Bloques 	Movimiento	Control
	Apariencia	Sensores
	Sonido	Operadores
	Lápiz	Variables
2.2 Pestañas Principales de Edición 	Programas	
	Disfraces	
	Sonidos	
2.3 Escenario 	Reproducir Archivo (con bandera verde)	
	Parar reproducción	
	Escenario	
3.1 Listado de funciones de Bloque 	Aparecen las funciones del Bloque seleccionado (ver ANEXO 2: LISTA DE FUNCIONES DE LOS BLOQUES EN SCRATCH)	
3.2 Área de Edición 	para la edición de Programas, Disfraces y Sonidos	
3.3 Objetos (Cast) 	Pintar Objeto	
	Elegir Objeto desde archivo	
	Objeto Sorpresa	
	Lista de Objetos	

Tabla 3: Opciones disponibles en las zonas en las que se divide la Interfaz.

III.I.III. PROGRAMACIÓN

Trabajando con Scratch, el alumnado adquiere importantes habilidades de pensamiento y aprende conceptos de computación. La tarea de programar requiere exclusivamente de habilidades de pensamiento claro y metódico.

Estos son algunos de los conceptos específicos de programación que se trabajan con Scratch:

Secuencia:
Para crear un programa en Scratch, se requiere pensar sistemáticamente sobre el orden de los pasos.

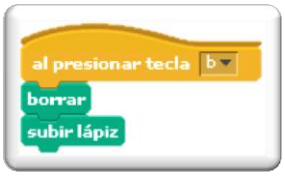
Iteración (ciclos):
Por siempre y *Repetir* se utilizan para crear iteraciones (repetición de una serie de instrucciones).

Variables:
Las variables sirven para almacenar números o cadenas de caracteres (palabras). Las instrucciones correspondientes a variables permiten crearlas y usarlas en un programa. Scratch admite tanto variables globales (para todos los objetos) como especificas para un solo objeto.

Sentencias Condicionales:
Si y *si-no* verifican si una proposición simple o compuesta es verdadera (si se cumple una condición).

Entradas vía teclado:
Preguntar y esperar solicita a los usuarios escribir algo mediante el teclado. *Respuesta* es una variable que almacena lo último que se ingresó vía teclado.

Manejo de eventos:
Al presionar tecla y *al presionar objeto* son ejemplos del manejo de eventos. Estas instrucciones de control responden a eventos provocados por el usuario o por otra parte del programa.



Hilos (paralelismo):
Poner en marcha dos pilas de instrucciones al mismo tiempo hace que se creen dos hilos independientes que se ejecutan en paralelo.

Números al azar:
La instrucción *número al azar entre* selecciona un número entero dentro de un rango dado.

Lógica booleana:

Las proposiciones compuestas se forman con dos o más proposiciones sencillas unidas por operadores lógicos (*y*, *o*, *no*). Por ejemplo: evaluar la

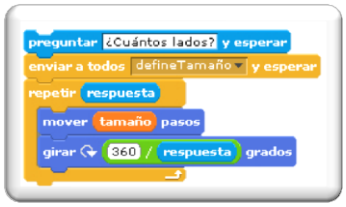
Diseño de interfaz de usuario:

Con *Scratch* se diseñan interfaces de usuario interactivas. Por ejemplo, usar objetos para que funcionen como botones. Por ejemplo, al hacer clic sobre el objeto "Lápiz" se ejecuta un conjunto de instrucciones.

Coordinación y sincronización:

La instrucción *enviar a todos* manda un mensaje a todos los Objetos y espera a que se ejecuten las acciones de los Objetos activados. La instrucción *al recibir* coordina acciones de diferentes objetos. Este par de instrucciones permiten la sincronización. Ejemplo:

Cuando un objeto *envía a todos defineTamaño*...



...entonces se ejecutan las instrucciones debajo de la instrucción *al recibir defineTamaño*

Listas (arreglos):

Las Listas son un tipo de estructura de datos que puede considerarse un arreglo bidimensional de "n x 1". Con varias listas se puede conformar una matriz (arreglo bidimensional de n x m).

Las instrucciones correspondientes a *listas* permiten almacenar y acceder arreglos de números o cadenas de caracteres.

Interacción dinámica:

Las instrucciones *x del ratón*, *y del ratón* y *volumen del sonido* se pueden utilizar como entrada dinámica para interactuar en tiempo real con los programas de Scratch.

III.I.IV. HERRAMIENTAS COMPLEMENTARIAS

Como se ha indicado con anterioridad, gracias a la versatilidad de *Scratch*, es posible aumentar su funcionalidad combinándolo con otras herramientas hardware y software de áreas como la informática (simuladores), la electrónica (placas electrónicas) y la robótica (robots y sensores).

A continuación se enumeran las herramientas TIC compatibles con *Scratch*:

Sensores y motor de Lego WeDo

Lego dispone de un motor, un sensor de inclinación y otro de distancia compatibles con *Scratch*. Estos elementos se conectan al ordenador mediante cable USB y el programa los reconoce automáticamente. El precio de cada sensor es de 15 euros aproximadamente.

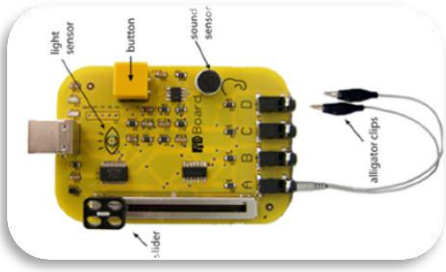
Al acceder al menú *Movimiento*, dispondremos de las funciones necesarias para controlar el motor a través del programa (*Imagen 2*): encender y apagar el motor, mantener el motor encendido durante un periodo de tiempo y determinar la dirección del motor. Si las funciones no están visibles, hay que acceder al menú *Editar* y pulsar la opción *Mostrar Bloques de Motor*.

Imagen 2: Funciones disponibles en Scratch para controlar el Motor de Lego WeDo.

En el caso de los sensores de inclinación y distancia, Scratch dispone dentro del menú *Sensores* de la función *valor del sensor_* para establecer el valor de los mismos (*Imagen 3*). El sensor de distancia tiene un rango de valores de 0 (punto mas cercano) a 100 (punto más lejano). El sensor de inclinación trabaja con un rango de valores de 0 a 4 (0: arriba, 1: abajo, 2: izquierda y 4: derecha).

Imagen 3: Función disponible en Scratch para establecer el valor de los sensores de inclinación y distancia de Lego Wedo.

Placa Electrónica PicoBoard



Esta placa compatible con Scratch, dispone de sensores de luz y sonido, botón (interruptor), deslizador, pinzas de contacto para medir resistencias y cuatro resistencias. La placa se conecta mediante un cable USB al ordenador.

El precio de la placa alcanza los 70 euros.

Scratch emplea la función *valor del sensor_* para establecer el valor de los sensores de luz y sonido, deslizador y las resistencias. Para determinar si el botón está presionado o si las resistencias están conectadas emplea la función *sensor_activado? (Imagen 4).* Esta función proporciona el valor *True* en caso de que el botón esté presionado o las resistencias conectadas y el valor *False* en caso contrario. Los sensores de luz y sonido, así como el

deslizador y las resistencias tienen un rango de valores entre 0 y 100 (valores mínimo y máximo respectivamente).

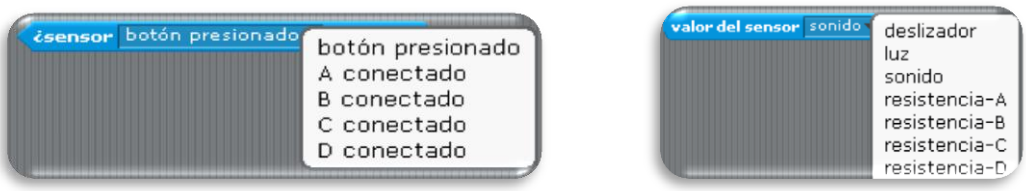


Imagen 4: Funciones disponibles en Scratch para establecer o determinar los valores de los elementos de la placa PicoBoard.

Para visualizar el valor de los elementos de la placa en el escenario, hacemos clic derecho sobre cualquiera de las funciones del bloque Movimiento, *valor del sensor_* o *¿sensor_activado?* y accedemos a la opción *mostrar variables de ScratchBoard* (*Imagen 5*):

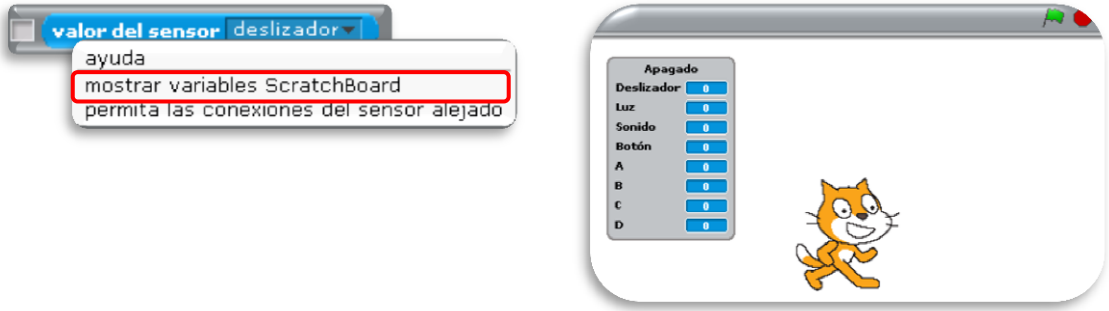


Imagen 5: Opción para visualizar el valor de los elementos de la placa PicoBoard (variables) a modo de tabla o lista en el escenario de Scratch.

A través de esta placa, Scratch consigue comunicarse con el mundo real y se pueden realizar proyectos más interactivos, haciendo uso de los datos provenientes de los elementos de la placa para controlar las animaciones. Por ejemplo, podemos realizar proyectos en los que un objeto emita un sonido cuando aplaudamos (sensor de sonido), el color del escenario se oscurezca al tapar el sensor de luz, un objeto se mueva en horizontal o vertical al manipular el deslizador etc.

Gracias a esta placa, el alumnado también puede reforzar la capacidad de recolección e interpretación de datos, en este caso, la información que proviene de la placa.

En el siguiente enlace podemos encontrar documentación sobre el funcionamiento de *Scratch* y *PicoBoard*, junto con algunas propuestas educativas:

https://sites.google.com/site/aprendoscratch/s30/picoboard#TOC-Algunos-ejemplos-de-proyectos-realizados-con-la-Picoboard

Existen en el mercado placas similares a PicoBoard, pero con mayores prestaciones. Por ejemplo la placa *Pico-Cat*, incorpora las siguientes mejoras:

- ✓ Joystick de 2 dimensiones (X,Y):

  Permite realizar programas en los que los objetos controlados se pueden mover por toda la pantalla
- ✓ Sensor de luz o detector de distancia:

  Además del receptor de infrarrojos (para determinar el nivel de luz existente) incorpora un emisor de infrarrojos que determina la distancia a la que se encuentran los objetos. El emisor emite luz no visible de modo permanente que rebota sobre los objetos que tiene

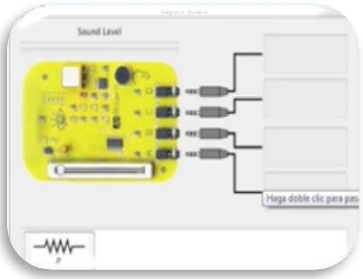
delante y es recibida por el receptor proporcionando valores en función de la distancia y el color de los objetos.

- ✓ Sensor de temperatura directamente calibrado en ºC: Es un tipo de semiconductor capaz de medir temperaturas de entre 0 y 60 ºC. El valor de este sensor se captura a través de la Variable C de Scratch.
- ✓ Entrada para un sensor externo: Se trata de una entrada para Jack de 3,5mm estéreo al que se pueden incorporar nuevos sensores externos para tener una entrada que permite una amplia gama de aplicaciones. Se incluye una guía didáctica de la autoconstrucción de un sensor destinado a la lectura de resistencia del cuerpo humano. Con esta aplicación es posible entre otros realizar sensores táctiles, etc.

En el siguiente enlace se puede encontrar información ampliada a cerca de esta placa:

http://www.tqlaboratorios.com/tqlab/index.php?page=shop.product_details&flypage=flypage_new.tpl&product_id=730&category_id=144&option=com_virtuemart&Itemid=101

Simulador de *PicoBoard*: *Rayure*



Herramienta de software libre que simula de manera virtual la placa electrónica PicoBoard. Su principal ventaja es que no supone coste alguno pero posibilita las mismas opciones que la placa *PicoBoard*.

De momento, solo está disponible para el sistema operativo MAC OS X. Las instrucciones para su instalación y la descarga del programa se pueden encontrar en el siguiente enlace: http://www.chaoticminds.org/rayure/

Una vez hemos instalado el simulador, para iniciar la placa virtual *Rayure* se debe seleccionar en la opción *Server Browser* la entrada que coincide con nuestro equipo y pulsar el botón *Connect* (*Imagen 6*).

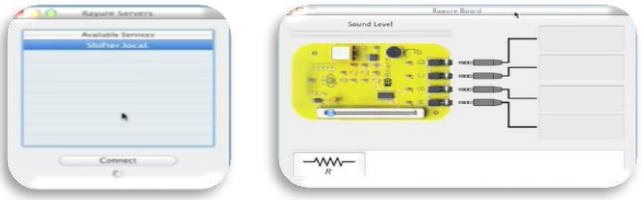


Imagen 6: Selección del servidor en el simulador que inicia la placa virtual Rayure.

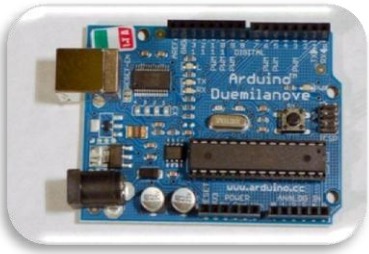
En *Scratch*, todas las aplicaciones locales de Rayure aparecen como puertos serie adicionales. Para acceder a esta lista, seleccionamos la opción anteriormente vista *mostrar variables de ScratchBoard* y estas aparecerán en la sección *Programas* (*Imagen 7):*

Imagen 7: Opción para mostrar en Scratch los valores de los sensores y elementos de la placa virtual Rayure a modo de lista.

A continuación hacemos clic izquierdo sobre la lista que aparece en el escenario, elegimos la opción *seleccionar puerto serial /USB* y seleccionamos el que coincide con el nombre de nuestro equipo (*Imagen 8*). En este momento ya podemos ver reflejadas las variaciones que realicemos sobre la placa virtual Rayure en el listado de parámetros visible en Scratch.

Imagen 8. Configuración de la comunicación entre la placa virtual Rayure y Scratch, a través del listado de sus parámetros.

Placa electrónica *Arduino*



Para emplear la placa electrónica Arduino, es necesario instalar la versión de *Scratch* denominada *SA4* (*Scratch for Arduino*). La placa tiene un precio de unos 30 euros pero dependiendo del modelo puede llegar a los 70 euros.

Las instrucciones de instalación y el archivo para la descarga de S4A están disponibles en el siguiente enlace:

http://seaside.citilab.eu/scratch/arduino

La placa cuenta con los siguientes componentes:

- Pin de referencia analógica
- Señal de tierra digital
- Pines digitales 3-13
- Pines digitales 1-2 / entrada y salida del puerto serie: TX/RX
- Botón de reset
- Entrada del circuito del programador serie
- Pines de entrada analógica 0-5
- Pines de alimentación y tierra
- Entrada de la fuente de alimentación externa (9-12V DC) – X1
- Conmuta entre fuente de alimentación externa o alimentación a través del puerto USB – SV1
- Puerto USB
-

S4A Incluye bloques complementarios a los presentes en Scratch para controlar sensores y actuadores conectados a *Arduino*. Cuenta asimismo con una tabla o lista de parámetros similar a la *PicoBoard* que informa del estado de los sensores (*Imagen 9*).

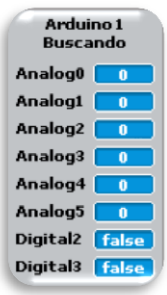


Imagen 9: Lista de parámetros en SA4 que muestra el estado de los sensores de la placa Arduino.

El programa *S4A* interactúa con Arduino enviando el estado de los actuadores y recibiendo el de los sensores cada 75 ms, por lo tanto el ancho de pulso ha de ser mayor que este período. La comunicación tiene lugar a través del protocolo que emplea la placa *PicoBoard* anteriormente descrita, el cual ya está implementado en la placa, dentro de un programa denominado firmware.

La configuración de entrada/salida está en proceso de desarrollo, por lo que por ahora los componentes tienen un modo concreto de conexión. Esta configuración ofrece lo siguiente:

- 6 entradas analógicas: pines analógicos 0-4.
- 2 entradas digitales: pines digitales 2 y 3
- 3 salidas analógicas: pines digitales 5, 6 y 9
- 3 salidas digitales: pines 10, 11 y 13
- 4 salidas especiales para conectar motores y servomotores de rotación continua *Parallax*: pines digitales 4, 7, 8 y 1.

Para conectar algunos sensores (LDR, potenciómetros, interruptores…) y algunos actuadores (motores, servomotores, leds, resistencias…) es necesario completar un circuito electrónico.

En *S4A* el control de las entradas y salidas analógicas/digitales, motores y servomotores se realiza a través de las funciones del bloque *Movimiento*.

A continuación se explican muestra las funciones disponibles en *S4A* para el control de los elementos dispuestos en la placa *Arduino*:

valor del sensor $_{nombre1}$ ➡ ***nombre1:*** *Analog0, Analog1, Analog2, Analog3, Analog4, Analog5, Digital2, Digital3*

Utilizando esta función, podemos leer los valores de las entradas analógicas y digitales (pines 0-4, 2 y 3). Los valores se muestran dentro de la lista de parámetros del escenario ($_{nombre1}$).

Las entradas analógicas de la placa tienen valores del rango [0-5] V y en el escenario se representan con valores del rango [0-1023].

Las entradas digitales en la placa están activadas o desactivadas y en el escenario se representan como True o False respectivamente.

¿sensor $_{nombre2}$ ***presionado?*** ➡ ***nombre2:*** *Digital2, Digital3*

Empleando esta función podemos leer el estado (encendido o apagado) de las entradas digitales (pines 2 y 3). Los estados se muestran dentro de a la lista de parámetros del escenario ($_{nombre2}$).

digital $_{número*}$ ***encendido / digital*** $_{número*}$ ***apagado*** ➡ ***número*:*** *10,11, 13.*

Estas dos funciones encienden o apagan las entradas digitales (pines 10,11 y 13) de la placa.

analógico *número** **valor** *valor** → ***número*:** *5,6, 9.* ***valor*:** *0-255.*

Con esta función se establecen los valores de las salidas analógicas (pines 5,6 y 9) de la placa. El rango de valores está entre 0 y 255 que equivale a 0 y 5 V respectivamente.

motor** número** **apagado → ***número*:** *4,7.*

Con esta función apagamos los motores ubicados en las salidas (pines 4 y 7).

***motor** número** **dirección** valor** → ***número*:** *4,7.* ***valor*:** *horario, antihorario.*

Con esta función establecemos la dirección (horaria o anti horaria) de los motores ubicados en las salidas de la placa (pines 4 y 7).

motor** número** **ángulo → ***valor número*:** *8,12.* ***valor*:** ***0-180°.***

Con esta función establecemos el ángulo (0-180°) de los servomotores ubicados en las salidas de la placa (pines 8 y 12).

SA4 también es compatible con las versiones de la placa *Arduino Duemilanove*, *Diecimila* y *Uno.*

En el siguiente enlace podemos encontrar uno de los muchos recursos disponibles en la red para iniciarse en el manejo de S4A y Arduino:

https://docs.google.com/file/d/0B4qM1H91FErrMDYwZTQzNjktMGE0NC00NmQ5LTllM2UtYjg4OTFkNDk2NTg4/edit?hl=en_US&pli=1

Kinect

Kinect es un dispositivo que permite controlar la consola Xbox 360 con nuestro cuerpo, realizando movimientos con la mano y convirtiendo todas nuestras partes del cuerpo en una extensión del personaje que aparece en pantalla.

Cuenta con una base giratoria motorizada, tres cámaras con sensor de movimiento, una cámara de captación de movimiento VGA con una resolución de 640 × 480 píxeles a 30FPS y una doble cámara de profundidad 3D de 640 × 480 píxeles a 30FPS. Además cuenta con cuatro micrófonos que pueden reconocer voces por separado.

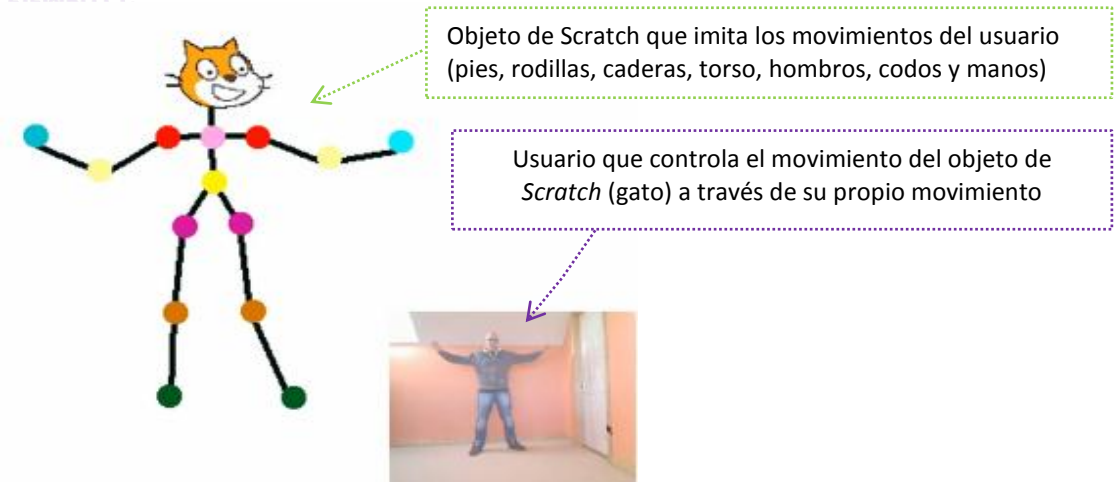
A pesar de las posibilidades que ofrece, tiene un precio bastante elevado, alrededor de 150 euros.

El programador Stephen Howells ha desarrollado un software (*Kinect2Scratch*) para controlar los objetos de Scratch a través de Kinect. Este software está disponible para sistemas operativos de Windows. Las instrucciones de instalación, el video demostrativo y el programa se pueden obtener en el siguiente enlace:

http://scratch.saorog.com/?page_id=2

Con el programa desarrollado por Stephen, los datos recopilados por la Kinect se envían a Scratch a través de un puerto remoto. Como resultado, nuestros movimientos se transforman en movimientos del personaje de Scratch (*Imagen 10*).

EJEMPLO 1:



EJEMPLO 2:

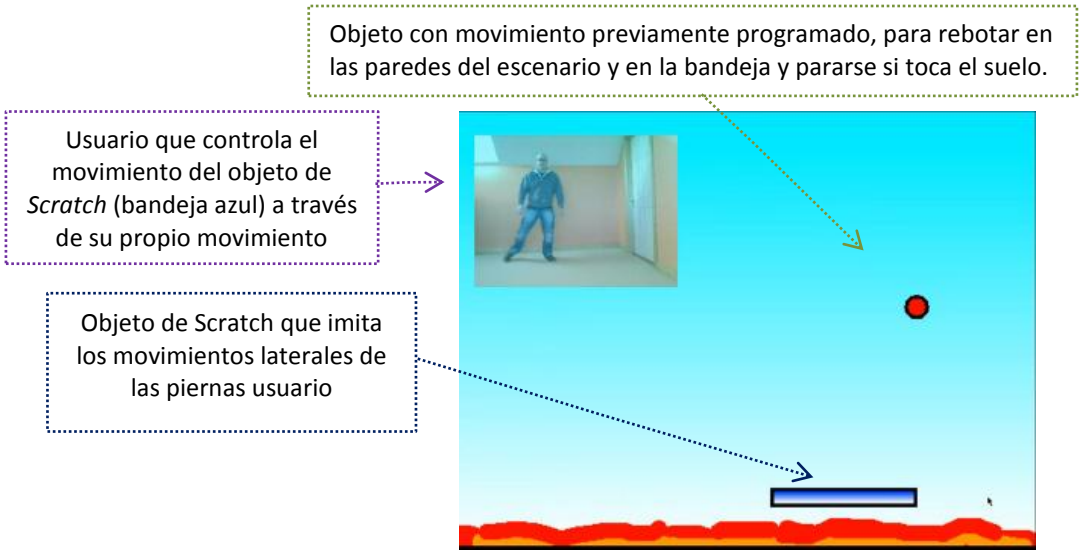


Imagen 10: Ilustración del funcionamiento de Scratch con Kinect a través del software Kinect2Scratch.

Sensores Virtuales

Al igual que ocurre en *Kinect2Scratch* es posible configurar sensores virtuales en *Scratch*, *Enchanting* y *S4A* a través de puertos de comunicación remotos, empleando la comunicación TCP/IP. Las tres aplicaciones pueden abrir una conexión en el puerto 42001 de la dirección IP del ordenador que está en uso. A través de esta conexión, las aplicaciones pueden recibir y enviar información a cualquier ordenador conectado a internet siguiendo un protocolo muy sencillo.

Para establecer la conexión debemos acceder al bloque *Sensores*, hacer clic derecho sobre cualquiera de las funciones *valor del sensor_* o *¿sensor _activo?* y seleccionar la opción *permita las conexiones del sensor alejado* (*Imagen 11*).

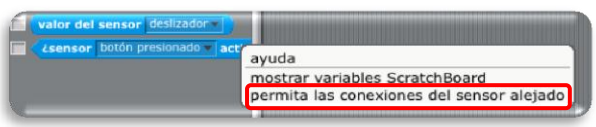


Imagen 11: Configuración del sensor virtual en Scratch, Enchanting y S4A.

A continuación hay que establecer una conexión TCP/IP con el puerto 42001 de la dirección IP del ordenador. En la pantalla que aparece, debemos escribir la siguiente instrucción:

sensor-update nombre[1] número[1]

Donde
- nombre[1]: nombre que le otorgamos al sensor virtual
- número[1]: valor numérico que le asignamos al sensor.

De este modo, se crea un sensor de nombre *nombre*[1] y valor *número*[1] que podemos visualizar en la aplicación haciendo clic derecho sobre cualquiera de las funciones *valor del sensor_* o *¿sensor_activado?* (*Imagen 12)*:

Imagen 12: Visualización del sensor virtual en Scratch, Enchanting y S4A.

Para visualizar el valor asignado a este sensor virtual en el Escenario, activamos la casilla izquierda de la función *valor del sensor_* (*Imagen 13*):

Imagen 13: Visualización del valor del sensor virtual en el Escenario de Scratch, Enchanting y S4A.

A través de esta conexión, es posible también la difusión de mensajes (*broadcast*) desde las aplicaciones *Scratch*, *S4A* o *Enchanting* o a los puertos remotos configurados. En este caso, se ha configurado el puerto 42001 y si nos conectamos con un Telnet a la IP del ordenador con el que estamos empleando la aplicación, veremos los mensajes que generemos. Para generar estos mensajes accedemos al bloque *Control*, clicamos sobre la función *enviar a todos_* y escribimos el mensaje que deseamos difundir en la ventana que se abre (*Imagen 14*):

Imagen 14: Difusión de mensajes (broadcast) hacia los puertos remotos en Scratch, Enchanting y S4A.

En la pantalla de Telnet observaremos el **mensaje** que hayamos escrito, detrás de la palabra broadcast:

```
telnet localhost 42001
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
broadcast "Prueba de mensaje"
```

No es posible enviar comandos de sensores virtuales ni Broadcast hacia las aplicaciones, ya que el protocolo obliga a que los dos primeros bytes del comando o mensaje indiquen el tamaño (en caracteres) del mensaje.

Existen múltiples aplicaciones que habilitando los puertos de comunicación remotos, hacen uso de sensores virtuales. A modo de ejemplo en el siguiente enlace podemos ver una aplicación capaz de convertir un *Smartphone Android* en un módulo DAQ (*Data AcQuisition*), enviando la información de sus sensores acelerómetro y compás y difundiendo dos estados (saltar y andar) a *Scratch*: http://smartphonedaq.com/scratch-sensor.page

III.II. *ENCHANTING* y ROBOTS *LEGO MINDSTORMS NXT*

III.II.I. ROBOTS LEGO MINDSTORMS

Para el desarrollo del bloque de contenidos de Control y Robótica, en los centros educativos normalmente se dispone del Kit para robótica *Lego Mindstorms* NXT. Los elementos que componen el Kit se describen a continuación:

A. El Cerebro del robot: Ladrillo NXT:

Para responder a los estímulos externos hay que tener la capacidad de tomar decisiones y para eso está el "cerebro" o controladora que en el caso del nuevo sistema de LEGO MINDSTORMS se llama NXT.

Es un microprocesador con puertos de entrada y salida y memoria para el almacenamiento de programas. Los puertos de salida se emplean para la conexión de los servomotores y los puertos de entrada se emplean para conectar los sensores. El ladrillo se comunica con el ordenador a través de un puerto USB o Bluetooth. En la imagen inferior (*Imagen 15*) se observan las partes del ladrillo:

Imagen 15: Componentes del Ladrillo NXT.

En el Menú Principal que se visualiza en la pantalla del ladrillo, se pueden ver los siguientes iconos:

					
Settings	**Try Me**	**My Files**	**NXT Program**	**View (Ver)**	**Bluetooth**
En esta sección puede cambiar los ajustes de sonido, el modo Sleep y eliminar archivos.	Una serie de programas de muestra para probar los distintos sensores.	Aquí es donde se guardan sus programas y sonidos.	Programa acciones sencillas en el NXT utilizando botones.	Ver todos los sensores conectados al NXT.	Localiza y se conecta a otros dispositivos Bluetooth.

B. Sistema Sensitivo del robot: Sensores

Una de las partes básicas de un robot es su sistema sensitivo, sin él no podría percibir su entorno, y en consecuencia, no podría responder ante los diferentes estímulos que se le presentan. Los elementos por medio de los que el robot recibe información de su entorno son los sensores que permiten medir distancias, iluminación del ambiente, niveles de ruido, temperatura etc.

Los sensores se acoplan al ladrillo a través de sus puertos de entrada, mediante un cable USB. En el Kit se incluyen los siguientes sensores:

Sensor de Tacto:

Al presionar y soltar el sensor, el robot identifica el tacto.

Sensor de Luz:

Es un sensor monocromático que distingue entre el blanco, el negro y la gama de grises existente entre ambos. La lectura de luz, la expresa en porcentaje.

Sensor de Sonido:

Este sensor detecta decibelios (dB) y decibelios ajustados (dBA), midiendo la presión sonora. Los decibelios ajustados, se ajustan a la audición humana. Los decibelios abarcan un rango de frecuencias más amplio, frecuencias inferiores y superiores a las detectables por el oído humano. Este sensor mide sonidos de un máximo de 90 dB. La lectura de la medida se expresa en porcentaje, de acuerdo con la siguiente tabla (*Tabla 4*):

Porcentaje	Descripción
4% - 5%	Habitación en silencio
5% - 10%	Personas hablando lejos
10% - 30%	Conversación normal
30% - 100%	Personas gritando y música a alto volumen

Tabla 4: Valores del sensor de sonido.

Sensor Ultrasónico:

A través de este sensor, el robot mide la distancia que hay hasta un objeto. Para calcular la distancia, el sensor emite un sonido y mide el tiempo que tarda esta señal en regresar. Usa el mismo principio que emplean los murciélagos y los sónares de las naves.

Tiene un rango de 0 a 255 cm con una precisión de ± 3 cm.

C. El sistema motor del robot: Motores y Servomotores

Además de sentir el robot ha de tener la capacidad responder a los estímulos externos, es decir, de hacer. De hecho es posible diseñar un robot sin sensores pero al contrario, no tiene ningún sentido que no sea capaz de hacer nada, aunque no sea más que generar sonidos.

Para hacer utiliza los motores con los que puede desplazarse, abrir y cerrar una pinza… o lo que se desee.

Servomotores:

El kit incorpora dos servomotores y los cables para conectarlos a los puertos de salida del ladrillo. Ambos servomotores se emplean para el desplazamiento del robot y uno queda libre para una posterior ampliación correspondiente al disparador de bolas, *ShooterBall*.

D. Esqueleto del robot: Piezas Technics y Mindstorms para el montaje del robot

Todo lo anterior necesita un esqueleto adaptado al uso que se le quiera dar que lo sustente. En función del objetivo deseado habrá que montar una estructura en forma de vehículo con ruedas, en forma de bípedo o cuadrúpedo, en forma de organizador de ladrillos LEGO en base a su color…

Sobre todo en el caso de los robots móviles es importante que esta estructura sea resistente, un robot móvil que puede chocar con una pared hay que montarlo de tal manera que no se desmonte durante su uso.

Tanto el set comercial de LEGO MINDSTORMS como el educativo ofrecen un conjunto de piezas suficiente para empezar a montar robots y con el tiempo puede completarse principalmente con la línea Lego Technics.

En la imagen izquierda se pueden observar las bolsas que contienen las 612 piezas Technics y 8 piezas Mindstorms que se incorporan en el Kit, con las que se pueden construir robots de diseños muy diversos.

E. Sus comportamientos

Pero ¿qué es un cerebro sin inteligencia? Nada, así que por medio de la programación hay que definir cómo se ha de comportar el robot ante los diversos estímulos externos.

Para ello Lego Mindstorms ofrece un software de programación gráfico conocido como NXT-G. Una serie de bloques ordenados de acuerdo con lo que se desea que haga el robot generan un programa que se transfiere al NXT para que éste, de modo totalmente autónomo sin necesitar el ordenador para nada, lo ejecute.

Sin embargo, existen más programas para la programación de robots NXT como los que se mencionan a continuación:

- **RobotC:**

Es un software dirigido a la educación, desarrollado por Robotics Academy, que permite programar en C. Es un software comercial de LEGO. La programación es textual y su entorno es propietario.

- **LabVieW Toolkit:**

Es un software de programación gráfica y posee un entorno propietario. Es posible programar el NXT directamente desde LabVIEW con el LabVIEW Toolkit for LEGO MINDSTORMS NXT, disponible en http://www.ni.com/academic/mindstorms/.

Permite una programación mucho más avanzada que con NXT-G, además, con el LabVIEW Toolkit for LEGO MINDSTORMS NXT es posible desarrollar nuevos bloques de programación para NXT-G.

- **NXC:**

Es un lenguaje de programación de alto nivel basado en C. Es un software de programación textual y de libre distribución (se puede descargar de http://bricxcc.sourceforge.net/).

- **leJOS NXJ:**

Es un software de programación textual y de libre distribución. leJOS NXJ facilita la programación del NXT con Java. Posee un completo firmware que sustituye el oficial de LEGO que funciona tanto en Windows como en Linux. Poco a poco se va completando e incorporando nuevos servicios. LeJOS proporciona, entre otros, los siguientes componentes: un firmware mejorado para el bloque NXT que incluye una máquina virtual Java (JVM), una API Java para usar el bloque NXT, comunicaciones con el PC y herramientas para cambiar el firmware, descargar programas, depurado de programas así como otras funciones.

En el siguiente apartado se describe este último software que permite la programación del robot a través del programa *Enchanting*, versión de Scratch para robótica.

III.II.II. INTRODUCCIÓN A ENCHANTING

Enchanting es una versión de *Scratch*, también libre, diseñada para la programación de los robots *Lego Mindstorms NXT* y *RCX*. A través de esta plataforma se pueden programar robots de Lego, sin coste alguno en el software.

Para poder controlar los robots a través de esta herramienta, es necesario instalar los siguientes programas:

- ***Software de Lego Mindstorms: Lego Mindstorms NXT.***
- ***Java JDK***
- ***LeJOS NXJ***
- ***Enchanting***

En el ANEXO 3: INSTALACIÓN DE *ENCHANTING*

se puede consultar el proceso de instalación completo.

El programa está orientado al control de los robots Lego, por lo que los proyectos que se pueden crear están ligados al ámbito de la robótica. Como veremos a continuación, en este caso las funciones del bloque Movimiento dependen de los motores de los que disponga nuestro NXT. Lo mismo sucede con las funciones del bloque Sensores que dependen de los sensores de los que dispone el robot.

Enchanting, a diferencia de Scratch, no ofrece la posibilidad de programar y diseñar animaciones con objetos, ya que el objetivo es el robot. Aun así, es visualizar en el escenario variables relacionadas con los sensores del robot y enviar señales de audio y de texto al robot, controlar el robot a través del teclado del ordenador etc.

III.II.III. INTERFAZ Y PROGRAMACIÓN DE *ENCHANTING*

Al ser una versión de S*cratch*, el interfaz en ambos programas es muy similar. La principal diferencia reside en que *Enchanting* exige una configuración inicial de los motores y sensores para hacer uso de estas funciones que en Scratch son distintas y aparecen por defecto en los bloques Movimiento y Motor respectivamente (*Imagen 16*):

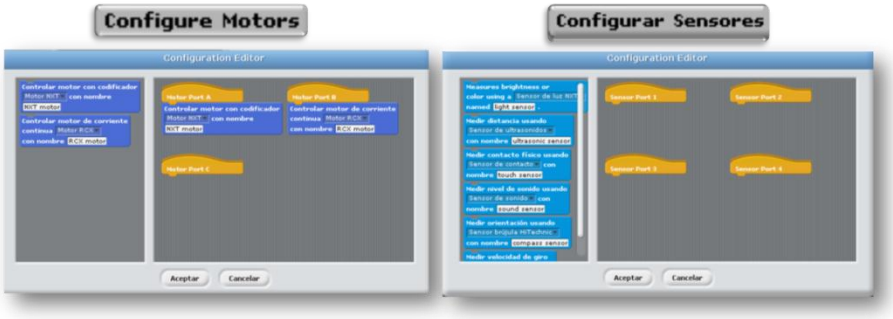


Imagen 16: Configuración inicial de motores y sensores en Enchanting.

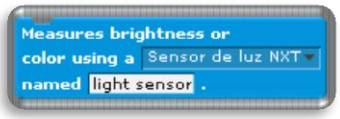
Para configurar los motores tipo NXT y RCX dentro del bloque Movimiento tras pulsar el botón Configure Motors, disponemos de las siguientes funciones (*Imagen 17*):

Imagen 17: Funciones para la configuración de motores NXT y RCX en Enchanting.

Una vez seleccionada la función, debemos arrastrarla hasta la parte derecha de la ventana y situarla en el puerto que tiene asignado (A, B o C) para que *Enchanting* pueda controlarlo (*Imagen 18*).

Imagen 18: Configuración de los puertos asignados a los motores NXT o RCX.

La configuración de los sensores del robot, se realiza del mismo modo pero en este caso accediendo al bloque Sensores y pulsando el botón *Configure Sensors*. *Enchanting* permite configurar los siguientes sensores:



Sensor de luz

(Mide el brillo o color)

Sensor de ultrasonidos

(Mide la distancia a un objeto)

Sensor de contacto

(Mide el contacto)

Sensor de sonido

(Mide el nivel de sonido)

Sensor brújula

(Mide la orientación)

Acelerómetro

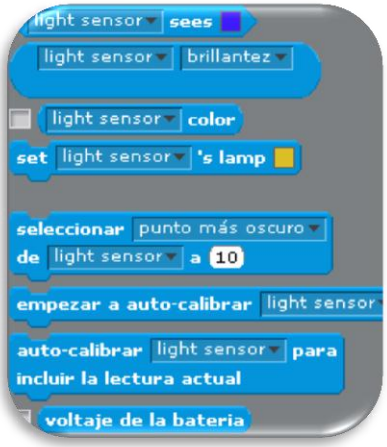
(Mide la velocidad de giro)

Tras seleccionar el sensor deseado, arrastramos el bloque a la parte derecha de la ventana, situándolo debajo del bloque correspondiente al puerto en el que hemos conectado el sensor, puerto 1, 2, 3 o 4 (*Imagen 19*):

Imagen 19: Configuración los puertos asignados a los sensores del robot Lego.

Una vez configurados los motores y sensores del robot, al acceder a los bloques *Movimiento* y *Sensores* visualizaremos las funciones disponibles para la configuración definida. La siguiente imagen (*Imagen 20*) muestra las funciones disponibles para cada sensor:

Funciones para el sensor de luz



Función para el sensor de distancia

Función para el sensor de tacto

Función para el sensor de sonido

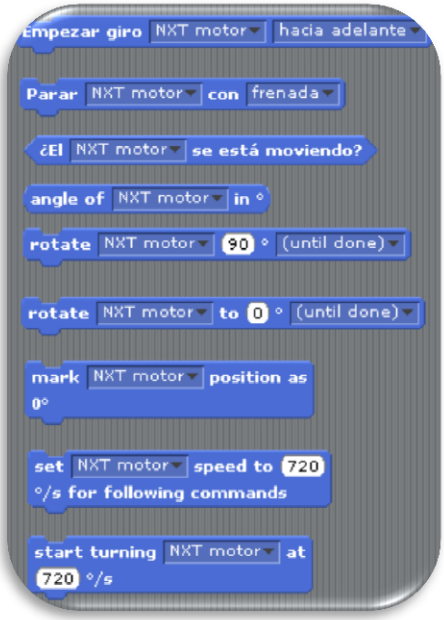
Funciones para el sensor brújula

Función para el acelerómetro

Imagen 20: Funciones disponibles en Enchanting para los sensores.

En la siguiente imagen (*Imagen 21*) podemos ver las funciones disponibles para los motores NXT y RCX:

Funciones para controlar el motor NXT con codificador



Funciones para controlar los motores de corriente continua RCX / NXT

Imagen 21: Funciones disponibles en Enchanting para controlar los motores NXT y RCX

La programación al igual que en *Scratch*, se realiza uniendo los bloques de las funciones de manera vertical. Nuevamente, se pueden trabajar conceptos de programación como:

- Secuencia
- Iteración (ciclos)
- Variables
- Sentencias Condicionales
- Entradas vía teclado
- Manejo de eventos
- Hilos (paralelismo)
- Números al azar
- Lógica booleana
- Coordinación y sincronización
- Listas (arreglos)

En este caso *Enchanting* no posibilita el diseño de interfaces interactivas, y en consecuencia, el escenario se convierte en un espacio para mostrar la información proveniente del robot a modo de lista o tabla de variables. Por esta razón, *Enchanting* no dispone del botón para ver el Escenario en modo presentación. Unido a esto, el bloque *Apariencia* cuenta únicamente con la función *imprimir_ (*para enviar texto al robot), desapareciendo las funciones relacionadas con el diseño de objetos (*cambiar el disfraz a_, cambiar efecto_por_* etc.).

El bloque de *Control*, es prácticamente el mismo que en Scratch. En este caso *Enchanting* también posibilita el control del robot con el teclado, a través de la función *wait until _button is pressed and released* pero sólo con las teclas *Intro*, *flecha izquierda*, *flecha derecha* y *Escape* (*Imagen 22*).

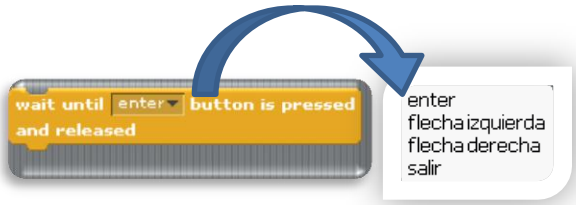


Imagen 22: Función para el control del robot en Enchanting mediante el teclado del ordenador.

Las funciones del bloque de *Sonido*, son algo limitadas en comparación a *Scratch*. Sin embargo, *Enchanting* dispone de tres sonidos (piano, xilófono y flauta) que pueden transmitirse al robot. También ofrece 25 notas diferentes para cada instrumento. Se puede controlar el volumen del sonido (rango 0-100) y el tiempo o ritmo del sonido en pulsos por minuto (rango 0-60). A continuación se muestran las funciones de sonido (*Imagen 23*):

Imagen 23: Funciones del bloque Sonido en Enchanting.

Las funciones del bloque *Operadores* son las mismas que en Scratch, aunque incluye alguna función que identifica entradas en un script o en un bloque y otra para clonar.

El bloque *Lápiz* no proporciona ninguna función, debido a que el programa está orientado al control del robot.

Por último, las funciones del bloque *Variables* se amplían en comparación a *Scratch*. Además de la posibilidad de crear variables y listas, *Enchanting* tiene la posibilidad de crear bloques. (*Imagen 24*). Podemos crear un bloque con el nombre y categoría que queramos (Movimiento, Control…) en el que añadimos las funciones que deseamos para el bloque.

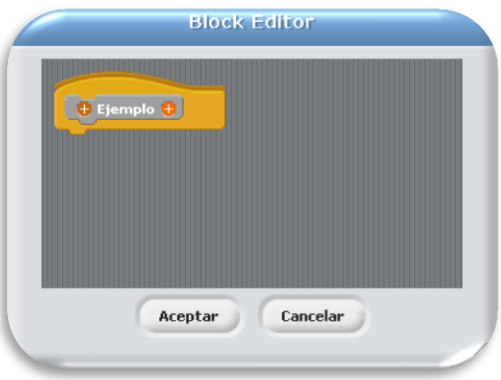


Imagen 24: Función para crear un bloque personalizado en Enchanting.

Además proporciona funciones para establecer/ cambiar/ borrar/ mostrar/ esconder el valor de las variables y añadir/borrar/ remplazar/contar elementos de las listas (*Imagen 25*):

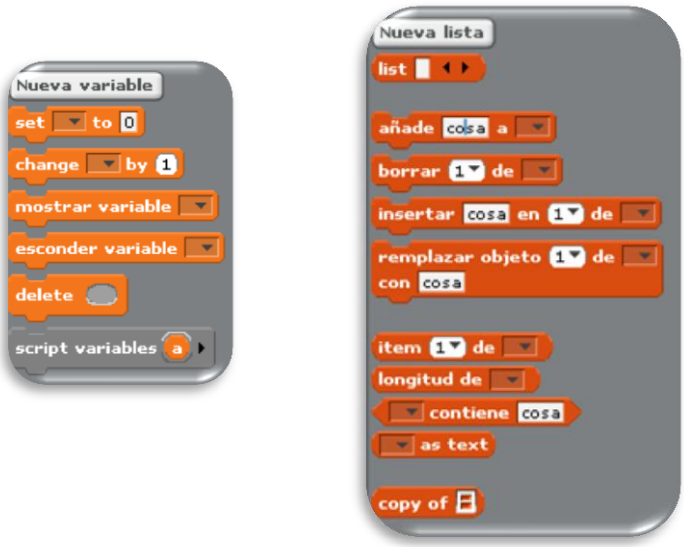


Imagen 25: Funciones disponibles para las variables y listas en Enchanting.

Para poder analizar las posibilidades de implementación en el aula de este software, en el siguiente capítulo se realiza una propuesta de implementación del bloque de contenidos de Control y Robótica haciendo uso de *Enchanting* y *Scratch*.

IV. PROPUESTA DE IMPLEMENTACIÓN DEL BLOQUE DE CONTENIDOS DE CONTROL Y ROBÓTICA CON *SCRATCH* Y *ENCHANTING*

En el capítulo anterior, se han explicado en detalle las múltiples posibilidades para el desarrollo de contenidos didácticos correspondientes a las áreas de informática, electrónica y robótica que ofrecen las plataformas *Scratch* y *Enchanting*.

El objetivo de este capítulo es facilitar el uso de estas herramientas dentro del aula a través del desarrollo de una propuesta didáctica centrada en el área de robótica. Para ello se realiza una propuesta de implementación orientada al bloque de contenidos de Control y Robótica del Currículum de Tecnología de 4º de E.S.O. Como se verá mas adelante, la metodología empleada para esta implementación es se basa en el Aprendizaje Basado en Proyectos.

IV.I. CONTENIDOS

Esta propuesta está centrada en el desarrollo del bloque de contenidos Control y Robótica definido en el Currículum para el área de Tecnología de 4º de E.S.O. La siguiente tabla recoge los contenidos didácticos de este bloque (*Tabla 5*):

AREA: TECNOLOGÍA	
NIVEL: 4º DE E.S.O	
BLOQUE DE CONTENIDOS	**CONTENIDOS DIDÁCTICOS**
CONTROL Y ROBÓTICA	Experimentación con sistemas automáticos, sensores, actuadores y aplicación de la realimentación en dispositivos de control.
	Diseño y construcción de robots.
	Uso del ordenador como elemento de programación y control. Trabajo con simuladores informáticos para verificar y comprobar el funcionamiento de los sistemas diseñados

Tabla 5: Bloque de Contenidos que se desarrolla en la propuesta: Currículum de Tecnología de 4º de E.S.O, Bloque de Control y Robótica.

IV.II. METODOLOGÍA

La metodología que se emplea en la propuesta es la de Aprendizaje Basado en Problemas, ya que es la más indicada para el desarrollo de contenidos didácticos del área de robótica educativa. A continuación se describen los aspectos de la metodología PBL, dando respuesta a una serie de preguntas a cerca de las características fundamentales de este método de trabajo:

¿Qué es PBL?

PBL o Aprendizaje Basado en Problemas, es una pedagogía centrada en el aprendizaje que se basa en las teorías actuales de aprendizaje incluyendo el constructivismo, constructivismo social y cognición situada. En las clases de PBL, al alumnado se le proponen problemas complejos de la vida real que han sido cuidadosamente elaborados para orientarlos hacia las

metas y objetivos del curso. El alumnado trabaja en grupos, bajo la supervisión de un facilitador, para resolver el problema. Durante el proceso de resolución, el alumnado aprende aspectos del trabajo en equipo, técnicas de resolución de problemas, técnicas de investigación, contenidos de la disciplina y comienza a pensar como un experto en de ese campo.

¿Cómo es un proceso PBL?

El siguiente diagrama (*Diagrama 1*) representa el ciclo de Aprendizaje Basado en Problemas cuyas fases se desarrollan a continuación:

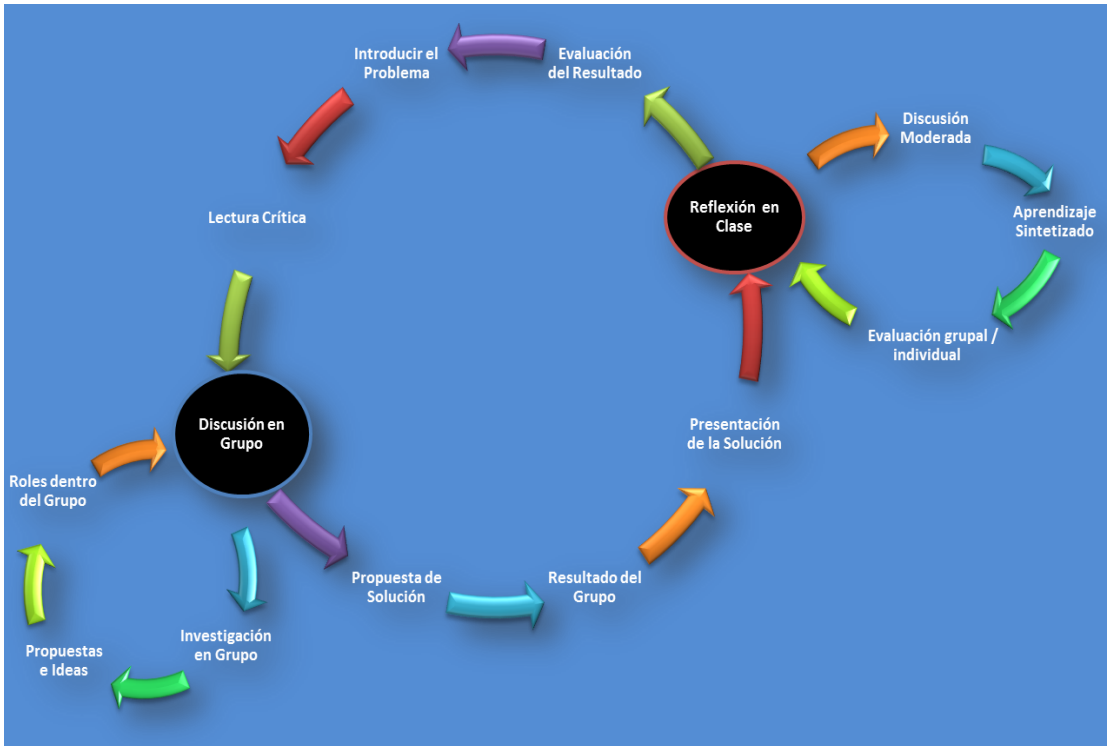


Diagrama 1: Ciclo del Aprendizaje Basado en Problemas (PBL)

- **Introducción del problema:**

Los problemas pueden introducirse de diversas maneras. Sin embargo, la clave está el diseño de la introducción del problema, que debe realizarse a través de un texto intrigante o provocativo, acaparando el interés del alumnado e involucrándolo en la acción del problema desde el inicio.

Normalmente se trata de un problema escrito. El instructor desglosa el problema y pide al alumnado que lea el problema por si mismo. A veces el problema se introduce en forma de video clip, artículo de periódico, pieza musical o la visita de un orador invitado. El formato no importa, la clave de un buen problema es su cuidada y meditada construcción.

- **Discusión en grupo:**

Tras la introducción y lectura individual, el alumnado lee el texto del problema con el grupo asignado e inicia las discusiones en grupo. Identifican los conocimientos que ya tienen y pueden serles útiles para completar la tarea de resolución del problema. Determinan entonces qué necesitan aprender para ser capaces de solucionar el problema (estas necesidades de saberes se suelen denominar cuestiones de aprendizaje). En este escenario, el alumnado probablemente elabore hipótesis preliminares sobre las soluciones al problema.

En una clase grande que emplea un modelo de facilitador rotante (multitud de grupos con un único instructor) a veces el facilitador reúne a todos los pequeños grupos para volver a formar

a la totalidad de la clase. La clase en su conjunto revisa las cuestiones de aprendizaje de cada pequeño grupo.

- **Roles del alumnado:**

Los instructores o miembros del grupo, pueden asignar roles a los y las estudiantes para facilitar el trabajo de grupo. Los roles incluyen al cronometrador, al abogado del diablo, al sintetizador y al interrogador.

- **Investigación:**

El alumnado examina una variedad de fuentes de información que probablemente contribuyan a la resolución del problema. El alumnado puede usar libros de texto, realizar búsquedas en literatura independiente, utilizar sitios web de calidad y/o pedir ayuda a bibliotecarios/as de referencia. La investigación se lleva a cabo sobre una base individual, sin embargo, pueden existir situaciones en las que los miembros del grupo dirijan su investigación en conjunto.

- **Discusión en Grupo:**

Tras esta fase de investigación inicial, el alumnado se reúne con los miembros del grupo para debatir sobre lo que han aprendido. La información se analiza e integra a medida que los miembros del grupo construyen nuevas comprensiones del problema e hipótesis sobre posibles soluciones. A medida que surgen nuevas preguntas, el ciclo de desarrollo de la investigación y los descubrimientos encontrados pueden repetirse.

- **Resolución del problema:**

El trabajo colaborativo del alumnado se traduce en un problema resuelto, tareas completadas y/o preguntas respondidas. En esta fase el alumnado probablemente presente un producto terminado para la evaluación o tal vez presente sus descubrimientos a la clase. Puede que también simplemente concluyan el problema y esperen demostrar sus nuevos conocimientos en un examen.

- **Finalización:**

Como parte del proceso de PBL, muchos instructores deciden concluir el trabajo con problemas de tipo resumen como, mini lecturas o grandes discusiones grupales. Estas experiencias proporcionan oportunidades adicionales al alumnado para pensar de manera crítica, es decir, aplicar, integrar, evaluar, analizar, y sintetizar la información.

- **Evaluación individual y grupal:**

La reflexión sobre el aprendizaje de uno mismo y las experiencias de grupo es un componente esencial del proceso PBL. Durante el transcurso de la reflexión sobre el contenido y el proceso dentro del ciclo de PBL, la reflexión sumativa sobre la contribución de los miembros del grupo posibilita al alumnado el desarrollo de sus habilidades para evaluar su propio rendimiento así como el de sus compañeros/as. Es más, las evaluaciones grupales que afectan a uno mismo, proporcionan incentivos adicionales al alumnado para ser participantes activos en el proceso colaborativo de resolución de problemas.

Normalmente se pide al alumnado que puntúe las actuaciones de los miembros del grupo con unos criterios específicos que se identifican en el modelo de evaluación de grupo. Al alumnado se le proporciona un resumen de los comentarios realizados por los miembros del grupo; sin embargo, los nombres de estos evaluadores no deben ser mostrados.

¿Cuáles son las estrategias efectivas de evaluación que apoyan y dirigen el aprendizaje?

En cursos PBL pueden emplearse con éxito varias formas distintas de evaluación:

- Las evaluaciones de grupo son también un buen modo de invitar a la persona que está aprendiendo a participar en el proceso de evaluación.
- Los proyectos ligados a los problemas, apoyados por una rúbrica, son una buen método de evaluación integrada en el curso.
- El alumnado puede presentar sus soluciones al problema. Nuevamente, estas presentaciones se pueden puntuar mediante una rúbrica.
- Los típicos test cuantitativos, incluyendo opciones múltiples, verdadero/ falso, y preguntas de respuesta breve no casan muy bien con la esencia de PBL, centrada en el aprendizaje. Sin embargo, son un buen método para comprobar el aprendizaje individual del alumnado en una clase. Es posible que aquellas disciplinas en las que los estudiantes deben realizar exámenes de regulación, quieran considerar el uso de test cuantitativos que a modo de exámenes regulados, aunque hayan utilizado PBL para la enseñanza.

IV.III. COMPETENCIAS BÁSICAS

La metodología PBL combinada con el bloque de contenidos de Control y Robótica se complementan muy bien para el desarrollo de las competencias básicas. A continuación se describe la contribución de la metodología PBL aplicada al área de robótica en cada una de las competencias básicas:

1. **Competencia en comunicación lingüística**

Durante un proceso de aprendizaje basado en PBL, esta competencia está presente de manera continua. Los grupos de alumnos/as deben discutir, debatir y argumentar las soluciones ideadas para la resolución de un problema, trabajando la expresión oral. Además, cada grupo debe analizar recursos como manuales, guías de usuario, vídeos prácticos o páginas web, recopilando, seleccionando, evaluando, sintetizando y restructurando estas informaciones para su aplicación en el problema planteado.

2. **Competencia matemática**

Esta competencia está muy ligada al PBL, ya que la resolución del problema se plantea desarrollando aspectos como la lógica, la abstracción etc. En el caso de la programación de los sensores empleados en robótica, el alumnado normalmente debe reflexionar en los sucesos derivados de valores detectados por los mismos (bucles condicionales). De otra parte, el alumnado trabaja aspectos de geometría como el radio y el perímetro de las ruedas, grados de una vuelta, aspectos de física como potencia, velocidad, distancia, tiempo, niveles sonoros, colorimetría etc. Las funciones de programación cuentan con variables relacionadas con los aspectos mencionados, que el alumnado define y analiza para la resolución del problema.

3. **Tratamiento de la información y competencia digital**

El PBL en al ámbito de la robótica desarrolla esta competencia mediante el análisis de objetos y sistemas tecnológicos y durante la resolución de problemas prácticos. Al analizar los componentes físicos del robot y las funciones del programa que lo gobierna el alumnado interpreta su funcionamiento. El ordenador se convierte en una herramienta fundamental para

la búsqueda de información, en el uso de las herramientas colaborativas de Internet, la utilización de software para la simulación o el diseño o la elaboración de documentos multimedia con herramientas ofimáticas para en la realización de la memoria y presentación del trabajo realizado al resto de la clase.

4. **Competencia social y ciudadana**

La metodología PBL fomenta fuertemente el desarrollo de esta competencia gracias al trabajo en equipo que plantea. El alumnado aprende a relacionarse en pequeños grupos, debiendo cooperar para alcanzar un objetivo común, asumir y delegar responsabilidades, trabajar aspectos como el respeto y la tolerancia etc. En definitiva, aprende a socializar y funcionar académica y personalmente con alumnos/as de diversa condición social, sexo, cultura, identidad y capacidad.

5. **Autonomía e iniciativa personal**

Esta competencia destaca sobre las demás en la metodología PBL, ya que para la resolución de los problemas planteados, el alumnado desarrolla estrategias propias con total autonomía, reduciendo al mínimo la intervención y supervisión del profesorado. De este modo, el alumnado realiza una reflexión personal sobre la mejor solución para el problema y experimenta por si mismo y de manera práctica la viabilidad de la misma. Durante este proceso, el alumnado detecta los problemas existentes y busca de manera autónoma los recursos necesarios para su superación.

6. Competencia de aprender a aprender

La metodología PBL trabaja esta competencia a través del planteamiento de un problema práctico. Para la búsqueda de soluciones, el alumnado aplica además de los conocimientos teóricos y prácticos propios del área, conocimientos derivados de experiencias personales de la vida real. Esto contribuye a que el alumnado experimente las fases que tienen lugar durante el aprendizaje a lo largo de la vida: análisis de la problemática, valoración de los recursos disponibles, búsqueda de recursos para alcanzar la solución, diseño de la solución inicial, evaluación práctica de el diseño inicial, detección de errores o aspectos mejorables, propuesta de mejora, comprobación de resultados, conclusiones del proceso. A lo largo de todas estas fases el alumnado completa un ciclo de aprendizaje muy amplio, en el que adquiere conocimientos didácticos de la materia, capacidades sociales y lingüísticas, pensamiento crítico, lógica aplicada a problemas prácticos y estrategias para la resolución de problemas de la vida real.

7**. Competencia cultural y artística**

El Aprendizaje Basado en Proyectos contribuye al desarrollo de esta competencia mediante la creatividad e imaginación que se trabajan para el diseño de soluciones óptimas a los problemas planteados. El alumnado abre su mente, analiza las experiencias personales y comparte distintos puntos de vista con los miembros del grupo para realizar el diseño de la solución más acertada. Del mismo modo, moldea el diseño inicial tras experimentar de primera mano la validez del mismo.

8. **Competencia en el conocimiento y la interacción con el mundo físico**

Esta competencia se pone en práctica a través de la manipulación de máquinas y objetos tecnológicos como el robot y el ordenador orientada a la solución de problemas prácticos presentes en el mundo real. El alumnado experimenta en primera persona problemas tecnológicos a los que el ser humano se enfrenta día a día. De este modo, conoce las necesidades tecnológicas de la sociedad y aprende a satisfacerlas ideando soluciones acordes a los recursos de los que dispone. Conceptos como eficacia, ahorro energético, sostenibilidad,

consumo, aprovechamiento de recursos naturales, respeto al medio ambiente están presentes durante un proyecto PBL en el área tecnológica.

IV.IV. HERRAMIENTAS Y RECURSOS DIDÁCTICOS

Para el desarrollo del bloque de Control y Robótica a través esta propuesta, se requieren las siguientes **Herramientas** TIC:

Software:

- *Scratch*
- Software de Lego Mindstorms: *Lego Mindstorms NXT.*
- *Java JDK*
- *LeJOS NXJ*
- *Enchanting*

La información para la descarga e instalación de los programas se puede consultar en el ANEXO 1: INSTALACIÓN DE SCRATCH

y en el ANEXO 3: INSTALACIÓN DE *ENCHANTING*

La información sobre el funcionamiento del software se describe en los Apartados SCTRATCH y ENCHANTING y *ROBOTS* LEGO *MINDSTORMS NXT*

Hardware:

- Kit de Lego Minstorms NXT
- Ordenadores (al menos un ordenador por cada cuatro alumnos).

La información para sobre el *Kit de Lego Mindstorms* se puede consultar en el Apartado III.II.I.

Además de estas herramientas, se hace uso de los siguientes **Recursos Didácticos:**

- **Libro de Tecnología de 4º de E.S.O (Anaya).**
- **Guía de Referencia de Scratch 1.4**

  La guía se puede descargar gratuitamente en el siguiente enlace: http://www.eduteka.org/pdfdir/ScratchGuiaReferencia.pdf

- **Blog de la Asignatura**

  Para esta Unidad Didáctica, se pretende hacer uso de un blog en el que el alumnado tenga a su disposición los recursos didácticos que se emplearán en el aula: resumen de las sesiones, ejemplos de programación, retos que deben llevar a cabo, etc.

IV.V. SECUENCIACIÓN

En este apartado se proponen 15 sesiones 55 minutos para la implementación del bloque de contenidos de Control y Robótica con *Scratch* y *Enchanting*. En la secuenciación, se describen una serie de actividades o retos a llevar a cabo por el alumnado, empleando la metodología PBL.

El contenido y desarrollo de las sesiones se describen en 9 tablas (*Tabla 6-Tabla 14*). La mayor parte de estas tablas describen el contenido de dos sesiones de 55 minutos (110 minutos en total). En las tablas se describen las actividades que se llevan a cabo durante las sesiones, así como los materiales y herramientas que se emplearán en las mismas (recursos didácticos, software, hardware etc.).

Los retos propuestos para algunas sesiones, se describen en el siguiente apartado (IV.V.I)

A modo de ejemplo, se adjuntan dos programas con los que se pueden desarrollar dos de los retos propuestos empleando las herramientas *Scratch* y *Enchanting* en el ANEXO 4: PROGRAMACIÓN CON SCRATCH DEL *RETO 4: VIAJE DE IDA Y VUELTA POR RECORRIDO DEFINIDO POR PARADAS*

y el ANEXO 5: PROGRAMACIÓN CON *ENCHANTING* DEL *RETO 6: ROBOT SIGUE LÍNEAS*

SESIÓN 1: INTRODUCCIÓN (55 Minutos)
ACTIVIDADES
Actividad 1: Presentación de la Unidad Didáctica Explicación teórica de la programación y evaluación de la Unidad Didáctica de Control y Robótica. Explicación de las herramientas que se emplearán a lo largo de la Unidad Didáctica: *Scratch* y *Enchanting*. Explicación de la metodología que se empleará en la Unidad Didáctica: Aprendizaje Basado en Problemas (PBL).
Actividad 2: Introducción Evolución de los sistemas automáticos. Explicación teórica de los siguientes conceptos • **Mecanización.** • **Automatización.** • **Robotización.**
Herramientas y Materiales
◆ Ficha de Programación de la Unidad Didáctica ◆ Ficha de Evaluación de la Unidad Didáctica ◆ Libro de Tecnología de 4º de E.S.O (Anaya)

Tabla 6: Sesión 1. Introducción.

SESIONES 2-3: INTRODUCCIÓN A LA PROGRAMACIÓN. INTRODUCCIÓN A SCRATCH. (110 Minutos)
ACTIVIDADES
Actividad 3: Introducción a la programación: Organigramas. Se realiza una explicación teórica de los Organigramas definiendo los siguientes conceptos: • Simbología • Tipos de estructuras: - Secuencial - Repetitiva ○ Mientras condición A hacer B ○ Repetir B hasta condición A ○ Hacer B hasta condición A • Alternativa - Si condición A hacer B1 sino hacer B2 - Hacer B1, B2… según C (dependiendo del valor de C se hace B1 o B2…)
Actividad 4: Creación de grupos Se divide al alumnado en grupos de 2 o 3 alumnos/as que se mantendrán durante las sesiones en las que se trabaje con Scratch. Los grupos serán heterogéneos, mezclando alumnos/as con buena disposición para el trabajo con los que resulten más problemáticos. Cada grupo dispondrá de un ordenador para trabajar con Scratch.
Actividad 5: Introducción a Scratch. Se reparte la Guía Básica de Scratch y se explican de manera práctica los siguientes aspectos: - Interface de Scratch - Programación por Bloques en Scratch - Objetos: Librería y Diseño - Bloque de Control - Bloque de Apariencia Se emplea un ejemplo de programación con los bloques de Control y Apariencia para iniciar al alumnado en la programación con Scratch. Se pide al alumnado que realice el organigrama correspondiente a los ejemplos.
Herramientas y Materiales
◆ Libro de Tecnología de 4º de E.S.O (Anaya) ◆ Guía de Referencia de Scratch. ◆ Blog de la Asignatura: Retos. ◆ Ordenadores (Software *Scratch*)

Tabla 7: Sesiones 2 y 3. Introducción a la programación. Introducción a Scratch.

SESIÓN 4: SISTEMAS DE CONTROL EN LAZO ABIERTO. (55 Minutos)
ACTIVIDADES
Actividad 6: Sistemas de Control en Lazo Abierto Explicación teórica un sistema de control en lazo abierto
Actividad 7: Programación de un Sistema de Control en Lazo Abierto Se pide al alumnado que realice el Reto 1: Cambio de disfraz controlado por teclado.
Herramientas y Materiales
◆ Libro de Tecnología de 4º de E.S.O (Anaya) ◆ Guía de Referencia de Scratch. ◆ Blog de la Asignatura: Retos. ◆ Ordenadores (Software Scratch)

Tabla 8: Sesión 4. Sistemas de Control en Lazo Abierto.

SESIÓN 5: SISTEMAS DE CONTROL EN LAZO CERRADO. (55 Minutos)
ACTIVIDADES
Actividad 8: Sistemas de Control en Lazo Cerrado Explicación teórica un sistema de control en lazo cerrado, poniendo ejemplos de la vida real (funcionamiento de una cisterna etc.).
Actividad 9: Bloques de Movimiento y Sensores Se explican los bloques de Movimiento y Sensores de Scratch a través de ejemplos prácticos.
Actividad 10: Programación de un sistema de Control en Lazo Cerrado (un controlador y un proceso) Se propone al alumnado el Reto 2: Viaje hacia delante sin parar
Herramientas y Materiales
◆ Libro de Tecnología de 4º de E.S.O (Anaya) ◆ Guía de Referencia de Scratch. ◆ Blog de la Asignatura: Retos. ◆ Ordenadores (Software Scratch)

Tabla 9: Sesión 5. Sistemas de Control en Lazo Cerrado.

SESIONES 6-7: SISTEMAS DE CONTROL EN LAZO CERRADO. (110 Minutos)
ACTIVIDADES
Actividad 11: Bloque de Sonido y Variables Se explica con ejemplos prácticos el funcionamiento de los bloques de *Sonidos* y *Variables* de Scratch.
Actividad 12: Programación de un sistema de Control en Lazo Cerrado (varios controladores y procesos) Se propone al alumnado el Reto 3: Viaje de ida y vuelta por recorrido definido por paradas
Herramientas y Materiales
◆ Guía de Referencia de Scratch. ◆ Blog de la Asignatura: Retos. ◆ Ordenadores (Software Scratch)

Tabla 10: Sesiones 6 y 7. Sistemas de Control en Lazo Cerrado.

SESIONES 8 Y 9: ARQUITECTURA DE ROBOTS. (110 Minutos)
ACTIVIDADES
Actividad 13: Introducción a la Arquitectura de los Robots NXT Se explican los conceptos básicos sobre la arquitectura de los robots NXT: • Cerebro: Ladrillo NXT • Sistema Sensitivo: Sensores • Esqueleto: Piezas *Technics* y *Lego* • Comportamiento: Programación
Actividad 14: Creación de nuevos grupos Se crean nuevos grupos, de 4-5 alumnos que se mantendrán hasta la finalización de esta Unidad Didáctica. Nuevamente, se crearán grupos heterogéneos mezclando alumnos/as con buena disposición al trabajo con aquellos/as que suelan mantener actitudes más reacias al trabajo.
Actividad 15: Montaje de un Robot Se reparte un *Kit de Lego Mindstorms* a cada grupo junto con el manual de instrucciones para montajes básicos. Se pide al alumnado que monte un robot básico.
Herramientas y Materiales
◆ Blog de la Asignatura. ◆ Manual de instrucciones del *Kit Lego Mindstorms NXT* ◆ Kits de *Lego Mindstorms NXT* (Motores NXT) ◆ Ordenadores.

Tabla 11: Sesiones 8 y 9. Arquitectura de Robots.

SESIONES 10 Y 11: INTRODUCCIÓN A LA PROGRAMACIÓN DE ROBOTS NXT CON *ENCHANTING*. (110 Minutos)
ACTIVIDADES
Actividad 16: Introducción al Programa *Enchanting.* Se explican las características elementales del programa *Enchanting*: • Interfaz • Configuración de Motores y Sensores • Bloques Se muestran sencillos ejemplos de programación del robot, empleando el bloque Movimiento de *Enchanting* y el Motor NXT.
Actividad 17: Programación del Movimiento de Avance y Retroceso para el Robot NXT Se pide al alumnado que realice el Reto 4: Robot Voy y Vengo. Para ello deberán modificar la arquitectura de los robots construidos en la pasada sesión y seleccionar los elementos necesarios para llevar a cabo el reto.
Herramientas y Materiales
◆ Blog de la Asignatura. ◆ Manual de instrucciones del *Kit Lego Mindstorms NXT* ◆ Kits de *Lego Mindstorms NXT (Motores NXT)* ◆ Ordenadores (Software *Enchanting*)

Tabla 12: Sesiones 10 y 11. Introducción a la Programación de Robots NXT con Enchanting.

SESIONES 12 Y 13: PROGRAMACIÓN CON *ENCHANTING* DE ROBOTS CON SENSORES (110 Minutos)
ACTIVIDADES
Actividad 18: Tipos de Sensores Se explica el funcionamiento básico de los siguientes sensores: • Sensor de Luz. • Sensor de Sonido. • Sensor de Ultrasonido. • Sensor de Tacto. Se explica la configuración de sensores en *Enchanting*. Se muestran sencillos ejemplos de programación del robot con sensores, empleando los bloques de Movimiento y Sensores.
Actividad 19: Programación del Movimiento de Avance y Retroceso Controlado del Robot NXT Se pide al alumnado que realice el Reto 5: Robot Voy y Vengo Cuando Detecto, modificando el programa del reto anterior. Para ello deberán modificar la arquitectura de los robots construidos en la pasada sesión y seleccionar los sensores necesarios para llevar a cabo el reto.
Herramientas y Materiales
◆ Blog de la Asignatura. ◆ Manual de instrucciones del *Kit Lego Mindstorms NXT* ◆ Kits de *Lego Mindstorms NXT* (Motores y Sensores NXT) ◆ Ordenadores (Software *Enchanting*)

Tabla 13: Sesiones 12 y 13. Programación con Enchanting de Robots con Sensores.

SESIONES 14 Y 15: PROGRAMACIÓN CON *ENCHANTING* DE ROBOTS SIGUE LINEAS (110 Minutos)
ACTIVIDADES
Actividad 20: Programación de un Robot Sigue Líneas Se propone al alumnado el Reto 6: Robot Sigue Líneas Para llevar a cabo el reto, el alumnado debe emplear el sensor de luz y los motores NXT.
Actividad 19: Competición de Robots Sigue Líneas Se pide al alumnado que realice el Reto 6:Competición de Robots Sigue Líneas Para ello el alumnado debe modificar el programa diseñado en el Reto 6, para tratar de lograr que el robot complete el recorrido en el menor tiempo posible.
Herramientas y Materiales
◆ Blog de la Asignatura. ◆ Manual de instrucciones del *Kit Lego Mindstorms NXT* ◆ Kits de *Lego Mindstorms NXT* (Motores y Sensores NXT) ◆ Ordenadores (Software *Enchanting*)

Tabla 14: Sesiones 14 y 15. Programación con Enchanting de Robots Sigue Líneas.

IV.V.I. RETOS

En este apartado se describen los retos propuestos para las sesiones:

◆ **Reto 1: Cambio de disfraz controlado por teclado**

El Alumnado debe diseñar un sistema de control en lazo abierto, en el que el controlador sea el teclado del ordenador y el proceso sea el cambio de disfraz en un objeto de diseño propio.

Por tanto, el alumnado debe conseguir que al pulsar determinadas teclas en el teclado del ordenador, el objeto de diseño personalizado cambie de disfraz.

◆ **Reto 2: Viaje hacia delante sin parar**

El alumnado debe diseñar un sistema de control en lazo cerrado, en el que el controlador sea la posición en el eje horizontal (x) del objeto y el proceso sea el movimiento horizontal

Es decir, el alumnado debe hacer que un objeto se mueva horizontalmente por el escenario y al llegar al borde del mismo, el objeto vuelva a su posición inicial. Este proceso debe ser infinito.

◆ **Reto 3: Viaje de ida y vuelta por recorrido definido por paradas**

El alumnado debe diseñar un sistema de control en lazo cerrado, definido por los siguientes elementos:

- **Controlador 1:** Posición (x) de las señales de parada
- Proceso 1.1: Movimiento del objeto por el recorrido definido por las paradas (ida y vuelta).
- Proceso 1.2: El objeto alerta emitiendo un sonido cuando llega a las señales de parada.
- Proceso 1.3: El objeto al llegar a cada señal de parada se detiene durante unos segundos.
- **Controlador 2:** Posición del objeto.
- Proceso 2.1: Las señales de parada alertan de que el autobús está en la parada cambiando de disfraz (disfraz 1 si no está en la parada, disfraz 2 si está en la parada).

En este caso, el alumnado en primer lugar debe diseñar unas posiciones para las señales de parada recorrido que definan el recorrido con coordenadas horizontales variables en (x).

En segundo lugar debe conseguir que el objeto se desplace por el recorrido definido por las señales de parada (ida y vuelta).

En tercer lugar el alumnado debe conseguir que cuando el objeto pase por las señales de parada el objeto emita un sonido y se detenga en ese punto durante unos segundos.

Por último, debe diseñar 2 disfraces para las señales de parada y conseguir que al paso del objeto las señales cambien de disfraz: disfraz 1 si no está en la parada, disfraz 2 está en la parada.

Una posible programación para este reto se puede consultar en el **ANEXO 4: PROGRAMACIÓN CON SCRATCH DEL *RETO 4: VIAJE DE IDA Y VUELTA POR RECORRIDO DEFINIDO POR PARADAS***

.

- **Reto 4: Robot Voy y Vengo.**

Se propone al alumnado que construyan un programa para que el robot avance una distancia y haga el camino de vuelta. Al completar el recorrido el robot deberá emitir un sonido de alerta.

- **Reto 5: Robot Voy y Vengo Cuando Detecto.**

Se propone al alumnado que modifiquen el programa del Reto 4 de manera que el robot avance una distancia y cuando un sensor reciba cierta información, espere unos segundos y haga el camino de vuelta. Al recibir esa información, el robot deberá emitir un sonido de alerta.

En función del sensor que escojan, la información que provoca el cambio de dirección del robot será distinta: detección de un color (sensor de luz), detección de un ruido (sensor de sonido), detección de un objeto (sensor de ultrasonido), detección de un contacto (sensor de contacto).

- **Reto 6: Robot Sigue Líneas**

Se propone al alumnado que programe el robot para que siga un recorrido definido por una línea negra, empleando cinta aislante de este color. El alumnado empleará el sensor de luz y los motores NXT para llevar a cabo este reto.

Una posible programación para este reto se puede consultar en el **ANEXO 5: PROGRAMACIÓN CON *ENCHANTING* DEL *RETO 6: ROBOT SIGUE LÍNEAS***

.

- **Reto 7: Competición de Robots Sigue Líneas**

Se propone al alumnado la realización de una competición por grupos de los robots construidos en el reto anterior.

El alumnado debe modificar el programa del reto anterior, para conseguir que el robot complete el recorrido en el menor tiempo posible.

Se premiará al robot más veloz y al más eficiente (el que sea más veloz y consuma menos batería).

IV.VI. EVALUACIÓN

Para evaluar al alumnado, se analizarán varios aspectos como, la actitud en las sesiones, la evaluación de grupo, los resultados obtenidos en la resolución de los retos y la elaboración de las fichas descriptivas del proceso de resolución de los retos.

La siguiente tabla (*Tabla 15*) muestra los documentos que se emplearán para la evaluación y la puntuación de cada apartado:

ASPECTOS	RECURSOS	%
1. Actitud	**Cuaderno de la Profesora**	**5%**
2. Trabajo en Grupo	**Ficha de Coevaluación de los miembros del grupo**	**15%**
3. Proceso de Resolución de los Retos	**Fichas Descriptivas de los Procesos de Resolución de los Retos**	**60%**
	Programas de los retos	
4. Resultados en la Resolución de los Retos.	**Fichas de Evaluación de los Retos**	**20%**

Tabla 15: Descripción de la Evaluación del Alumnado.

A continuación se describe cada uno de los apartados de la evaluación:

1. Actitud

Para evaluar la actitud se emplea el Cuaderno de la Profesora y supone un total de 0.5 puntos sobre 10 de la evaluación final de la Unidad Didáctica. La profesora analizará la actitud de cada alumno a lo largo de toda la Unidad Didáctica, teniendo en cuenta los siguientes aspectos:

- **Cumplimiento de las responsabilidades de grupo (0.25 ptos):**
  Trabaja en equipo siempre: 0.25 puntos.
  Trabaja en equipo a veces: 0.15 puntos.
  No trabaja en equipo: 0 puntos.

- **Cumplimiento de las Normas del Taller (0.25 ptos):**
  Recoge el material: 0.10 puntos.
  Cuida el material: 0.15 puntos.

2. Trabajo en Grupo

Para evaluar este apartado, al final de la Unidad Didáctica se repartirá a cada alumno/a una Ficha de Coevaluación de los miembros del Grupo. Este apartado supone un total de 1.5 puntos sobre 10 de la evaluación final de la Unidad Didáctica. En ella, cada alumno/a debe puntuar la actitud de los miembros del grupo, completando la siguiente Ficha (*Ficha 1*):

FICHA DE COEVALUACIÓN DE LOS MIEMBROS DEL GRUPO	
Unidad Didáctica	
Nombre y Apellidos:	
Grupo:	

Escribe los nombres de los miembros de tu grupo y evalúa de manera global el trabajo y la actitud de cada uno, otorgando puntuaciones de 1 a 5:

Miembros	Nombre y Apellidos	Nota Final (1-5)
Estudiante 1		
Estudiante 2		
Estudiante 3		
Estudiante 4		

Evalúa el trabajo y la actitud de cada uno de los miembros del grupo, puntuando de 1 a 5 los siguientes aspectos:

Aspectos	Estudiante 1	Estudiante 2	Estudiante 3
Ha cumplido con sus responsabilidades.			
Ha completado a tiempo sus responsabilidades.			
Ha contribuido a la resolución de los retos con aportaciones e ideas personales.			
Ha mostrado interés por el trabajo de grupo, manteniendo una actitud proactiva.			
Ha mostrado una actitud tolerante y colaborativa con el resto de miembros del grupo.			
Nota Final			

Ficha 1: Ficha de Coevaluación de los Miembros del Grupo.

3. Proceso de Resolución de los Retos

Para evaluar este apartado, al final de cada reto cada grupo debe entregar a la profesora una Ficha Descriptiva del Proceso de Resolución del Reto junto con el archivo del programa. La ficha es una memoria que resume el proceso de resolución de cada reto y junto con el archivo del programa supone 6 puntos sobre 10 de la evaluación de la Unidad Didáctica A continuación se expone un modelo de esta ficha (*Ficha 2*)

FICHA DESCRIPTIVA DEL PROCESO DE RESOLUCIÓN DEL RETO			
Unidad Didáctica			
Grupo:			
Nº Reto			
Dibuja el Diagrama de Bloques correspondiente al este reto:			
Enumera los sensores y motores empleados, describiendo su función			
Sensores / Motores	**Función**		
Nombra los bloques de programación y las funciones empleadas y describe su utilidad			
Bloque de Programación	**Función**	**Utilidad**	
Responde a las siguientes cuestiones:			
¿Se ha superado el reto?			
Objetivos Alcanzados:			
Dificultades Encontradas			
Resolución de las Dificultades:			
Recursos para la Resolución del Reto:			

Ficha 2: Ficha Descriptiva del Proceso de Resolución del Reto

4. Resultados en la Resolución de los Retos.

Para evaluar este apartado, la profesora puntuará los resultados de cada grupo, en la resolución de los retos empleando una Ficha de Evaluación de los Retos. Este apartado supone 2 puntos sobre 10 en la evaluación final de la Unidad Didáctica. En esta Ficha se evalúan los siguientes aspectos (*Ficha 3*):

FICHA DE EVALUACIÓN DEL RETO	
Unidad Didáctica	
Grupo:	
Nº Reto	

Cumplimiento de los objetivos propuestos en el reto (1 punto):

Cumplimento de Objetivos	Puntuación
Se han alcanzado todos los objetivos.	1 punto
Se han alcanzado la mayoría de los objetivos.	0.75 puntos
Se ha alcanzado la mitad de los objetivos.	0.5 puntos
Se han alcanzado menos de la mitad de los objetivos.	0.25 puntos
No se ha alcanzado ninguno de los objetivos.	0 puntos

Calidad del Resultado (1 punto):

Calidad del Resultado	Puntuación
El resultado es excelente.	1 punto.
El resultado es notable.	0.75 puntos
El resultado es adecuado.	0.5 puntos.
El resultado es algo insuficiente.	0.25 puntos.
El resultado es insuficiente.	0 puntos.

Ficha 3: Ficha de Evaluación del Reto.

ANEXOS

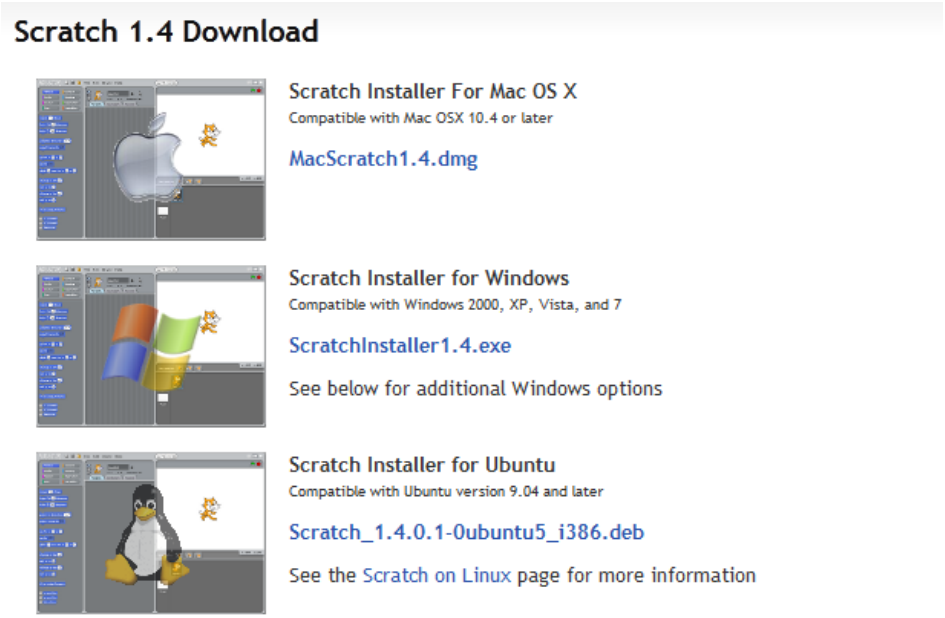
ANEXO 1: INSTALACIÓN DE SCRATCH

Para instalar Scratch, en primer lugar accedemos a la página oficial:

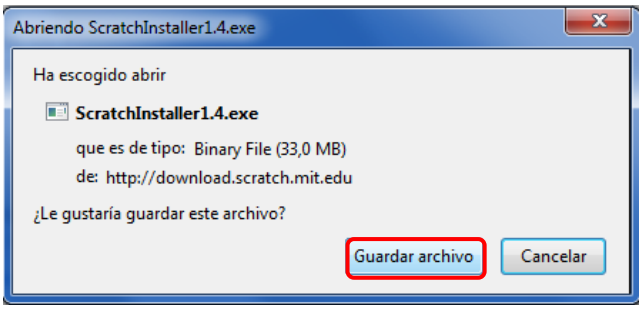
http://scratch.mit.edu/

A continuación pulsamos el botón ***Download Scratch***:

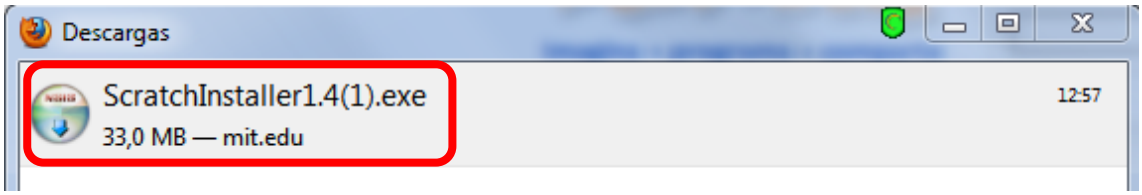
Después, descargamos el programa pulsando el enlace que se ajuste al sistema operativo que tenemos instalado: Mac OS X, Windows (2000, XP, Vista y 7) o Ubuntu:



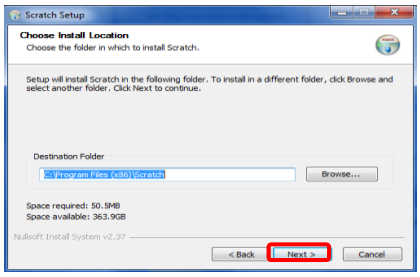
Tras pulsar el enlace correspondiente se abre la ventana de descarga del programa, en la que debemos pulsar el botón ***Guardar archivo***:

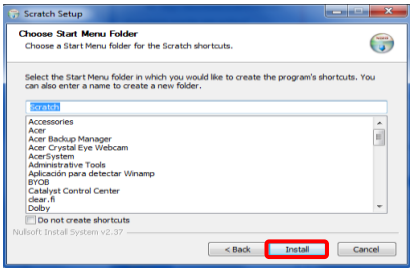


Después debemos abrimos el **ejecutable** descargado:

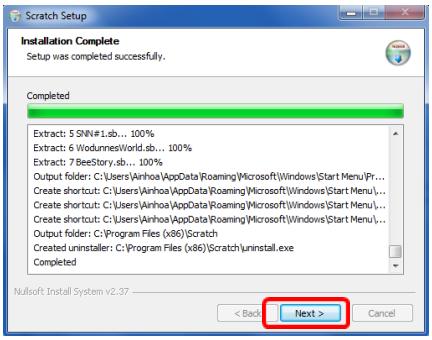


La instalación se iniciará y deberemos ir pulsando los botones ***Next*** para continuar el proceso, en las sucesivas ventanas que aparezcan:





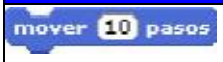
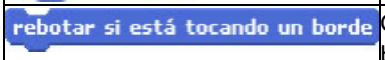
En este momento, aparecerá una ventana indicando la evolución de la instalación y al completarse debemos nuevamente pulsar el botón ***Next*** para que el proceso continúe:



Para terminar, pulsamos el botón ***Finish*** para finalizar el proceso**;** el programa quedará instalado en el ordenador y se abrirá de modo automático.

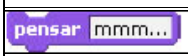
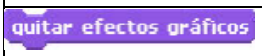
ANEXO 2: LISTA DE FUNCIONES DE LOS BLOQUES EN *SCRATCH*

Los bloques de *Scratch* están organizados dentro de ocho categorías de códigos de color: Movimiento, Apariencia, Sonido, Lápiz, Control, Sensores, Operadores y Variables. Las funciones correspondientes a cada bloque se resumen en las tablas expuestas a continuación:

MOVIMIENTO	
	Mueve el Objeto hacia adelante o hacia atrás.
	Rota el Objeto en el sentido de las manecillas del reloj.
	Rota el Objeto en el sentido contrario a las manecillas del reloj.
	Apunta el Objeto en la dirección especificada (0=arriba; 90=derecha; 180=abajo; -90=izquierda).
	Apunta el Objeto hacia el puntero del ratón o hacia otro Objeto.
	Mueve el Objeto hacia una posición específica de X, Y en el escenario.
	Mueve el Objeto a la ubicación del puntero del ratón o de otro Objeto.
	Mueve el Objeto suavemente a una posición determinada en un lapso de tiempo específico.
	Cambia la posición X del Objeto en una cantidad determinada (incrementa)
	Fija la posición X del Objeto a un valor específico.
	Modifica la posición Y del Objeto en una cantidad determinada (incrementa)
	Fija la posición Y del Objeto a un valor específico.
	Gira el Objeto en sentido contrario, cuando este toca un borde del escenario.
	Informa la posición X del Objeto. (Rango entre –240 a 240)
	Informa la posición Y del Objeto (Rango entre -180 a 180)
	Informa la dirección del Objeto (0=arriba; 90=derecha; -90=izquierda;180=abajo)

BLOQUES DE MOTOR: Los Bloques de Motor solo aparecen si se selecciona *Mostrar Bloques de Motor* en el menú *Editar* o se conecta un *LEGO WeDo*.

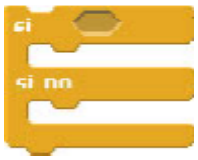
	Enciende el motor por un período de tiempo determinado
	Enciende el motor
	Apaga el motor
	Fija el poder o capacidad del motor y lo enciende. (Rango entre 0 y 100)
	Establece o modifica la dirección del motor, pero no lo enciende. (en esta dirección = sentido manecillas del reloj; en esta otra dirección = contrario a las manecillas del reloj; reversa = cambio de dirección)

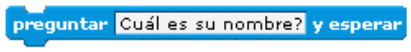
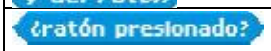
APARIENCIA	
	Modifica la apariencia del Objeto cambiando de disfraz.
	Cambia el disfraz del Objeto por el siguiente disfraz en la lista de disfraces (cuando llega al final del listado de estos, vuelve a comenzar con el primer disfraz).
	Informa el número correspondiente al disfraz actual del Objeto.
	Modifica la apariencia del escenario cambiando a un fondo diferente.
	Modifica la apariencia del escenario pasando al siguiente fondo disponible en el listado de estos.
	Reporta el número del fondo actual del escenario.
	Despliega una nube de diálogo del Objeto durante un lapso de tiempo determinado
	Despliega una nube de diálogo del Objeto (se puede eliminar esta burbuja de diálogo ejecutando este
	Despliega una nube de pensamiento del Objeto durante un determinado lapso de
	Despliega una nube de pensamiento del Objeto.
	Modifica (incrementa o decrementa) un efecto visual del Objeto en una cantidad especificada (use el menú desplegable para seleccionar el efecto).
	Establece un efecto visual a un número dado (la mayoría de efectos visuales se ubica en un rango de 0
	Limpia o borra todos los efectos gráficos de un Objeto
	Modifica el tamaño del Objeto en una cantidad especificada (incrementa o decrementa).
	Ajusta el tamaño del Objeto en un porcentaje (%) específico respecto a su tamaño original.
	Informa el tamaño del Objeto como porcentaje (%) de su tamaño original.
	Hace aparecer un Objeto en el escenario.
	Hace desaparecer un Objeto del escenario (cuando el Objeto está escondido, otros Objetos no lo pueden detectar con el bloque "¿tocando?").
	Ubica el Objeto al frente de todos los demás Objetos (capa superior).
	Mueve el Objeto hacia atrás, un número determinado de capas, de manera que pueda ocultarse detrás de otros Objetos.

SONIDO	
tocar sonido miau	Comienza la reproducción del sonido seleccionado del menú desplegable, e inmediatamente pasa al siguiente bloque aunque el sonido se esté ejecutando aún.
tocar sonido miau y esperar	Reproduce un sonido y espera hasta que el sonido termine, antes de continuar con el bloque siguiente.
detener todos los sonidos	Detiene todos los sonidos.
tocar tambor 48 durante 0.2 pulsos	Reproduce un determinado número de sonido de tambor, seleccionado del menú desplegable, durante un número específico de pulsos.
	Reproduce una nota musical (número altos para tonos altos) durante un número específico de pulsos.
silencio por 0.2 pulsos	Descansa, no toca nada, durante un número específico de pulsos.
fijar instrumento a 1	Establece el tipo de instrumento que usa el Objeto para los bloques de
cambiar volumen por -10	Modifica el volumen del sonido del Objeto en un valor especificado
fijar volumen a 100	Fija el volumen del sonido del Objeto a un valor especifico.
volumen	Informa el volumen del sonido del Objeto.
cambiar tempo por 20	Modifica el tempo del Objeto en una cantidad específica (incrementa o decrementa).
fijar tempo a 60 ppm	Fija el tempo del Objeto a un valor especificado de pulsos por minuto.
tempo	Informa el tempo del Objeto en pulsos por minuto.

LÁPIZ	
borrar	Borra todas las marcas de lápiz y de sellos (estampados) del Escenario.
bajar lápiz	Baja el lápiz del Objeto, de manera que este pinte a medida que se mueve.
subir lápiz	Levanta el lápiz del Objeto, de manera que no pinte cuando se mueva.
fijar color de lápiz a	Establece el color del lápiz, basado en la selección hecha en la paleta de color.
cambiar color del lápiz por 10	Modifica el color del lápiz en una cantidad específica (incrementa o decrementa).
fijar color de lápiz a 0	Establece el color del lápiz a un valor determinado. (color-lápiz=0 en el borde rojo del arco iris; color-lápiz=100 en el borde azul del arco iris. Rango de 0 a 200
cambiar intensidad de lápiz por 10	Modifica la intensidad del lápiz en una cantidad especificada(incrementa o decrementa).
fijar intensidad de lápiz a 50	Establece un valor específico para la intensidad del lápiz (sombra- lápiz=0 es muy oscura; sombra-lápiz=100 es muy clara. El valor por defecto es 50, a menos que se
cambiar tamaño de lápiz por 1	Cambia el grosor del lápiz en una cantidad específica (incrementa o decrementa en una cantidad específica).

	Establece el grosor del lápiz.
	Estampa o copia la imagen del Objeto en el Escenario.

CONTROL	
	Ejecuta el programa que tiene debajo al hacer clic en la bandera verde.
	Ejecuta el programa que tiene debajo al presionar una tecla específica.
	Ejecuta el programa que tiene debajo al hacer clic en un Objeto.
	Espera un número determinado de segundos y continúa luego con el bloque siguiente.
	Ejecuta continuamente los bloques en su interior.
	Ejecuta, un número específico de veces, los bloques en su interior.
	Envía un mensaje a todos los Objetos y luego continúa con el bloque siguiente sin esperar a que se realicen las acciones de los Objetos activados.
	Envía un mensaje a todos los Objetos, activándolos para que hagan algo y espera a que todos terminen antes de continuar con el siguiente bloque.
	Ejecuta el programa que tiene debajo cuando recibe un mensaje específico "enviar a todos" (Broadcast).
	Comprueba continuamente si una condición es verdadera; cada que es verdadera, ejecuta los bloques en su interior.
	Si la condición es verdadera, ejecuta los bloques en su interior.
	Si la condición es verdadera, ejecuta los bloques dentro de la porciónsi; si no, ejecuta los bloques que están dentro de la porción si no.
	Espera hasta que la condición sea verdadera, para ejecutar los bloques siguientes.
	Comprueba si la condición es falsa; si lo es, ejecuta los bloques en su interior y vuelve a chequear la condición. Si la condición es verdadera, pasa a los bloques siguientes.
	Detiene el programa (que se está ejecutando dentro de un Objeto)
	Detiene todos los programas de todos los Objetos.

SENSORES	
	Informa verdadero, si el Objeto está tocando un Objeto específico, un borde o el puntero del ratón (seleccionados del menú desplegable).
	Informa verdadero, si el Objeto está tocando un color específico. (Haga clic en la paleta de color y luego utilice el gotero para seleccionar el color).
	Reporta verdadero si el primer color (dentro del Objeto), está tocando un segundo color (tanto en el fondo como en otro Objeto). Haga clic en la paleta de color y luego utilice el gotero para seleccionar el color.
	Formula una pregunta en la pantalla y guarda lo que se ingresa por teclado en la  . Hace que el programa espere hasta que se presione la tecla
	Reporta la entrada de teclado, del uso más reciente de  Se comparte para todos los Objetos (Global)
	Informa la posición "X" del puntero del ratón.
	Informa la posición "Y" del puntero del ratón.
	Informa verdadero, si el botón del ratón está presionado.
	Informa verdadero, si una tecla específica está presionada.
	Informa la distancia desde un Objeto específico o desde el puntero del ratón.
	Fija el cronómetro en 0.
	Reporta el valor del cronometro en segundos (el cronómetro siempre está contando).
	Informa una propiedad o variable de otro Objeto.
	Reporta el volumen de los sonidos captados por el micrófono del computador (entre 1 y 100).
	Reporta verdadero, si el volumen del sonido captado por el micrófono del computador es mayor de 30 (en escala de 1 a 100).
	Informa el valor de un sensor específico. Para usar este bloque se necesita tener un sensor conectado a su computador. Puede usar esto con una tarjeta de
	Informa verdadero, si un sensor específico está presionado. Para usar este bloque se necesita tener una tarjeta de sensores para *Scratch* conectado a su

OPERADORES	
	Suma dos números.
	Resta dos números (Sustrae el segundo número de el primero)
	Multiplica dos números.
	Divide dos números (Divide el primer número entre el segundo)
	Selecciona al azar un número entero dentro de un rango especificado.
	Informa verdadero, si el primer valor es menor que el segundo
	Reporta verdadero, si dos valores son iguales.
	Informa verdadero, si el primer valor es mayor que el
	Informa verdadero, si ambas condiciones son
	Informa verdadero, si una de las dos condiciones es
	Reporta verdadero, si la condición es falsa; reporta falso si la condición es verdadera.
	Concatena (combina) cadenas de letras (caracteres)
	Informa el número de letras en una cadena
	Informa la letra en una posición específica dentro de una cadena
	Reporta el resultado de una función seleccionada (abs, raíz cuadrada, sin, cos, tan, asin, acos, atan, ln, log, e^, 10^) aplicada a un número específico.
	Informa el residuo(módulo) de la división del primer número entre el segundo número.
	Informa el entero más cercano a un número.

VARIABLES	
	Permite crear y nombrar una nueva variable. Cuando usted crea una variable, aparecen los bloques correspondientes a ella. Se puede escoger si la variable es para todos los Objetos (global) o solo para un Objeto (local)
	Borra todos los bloques asociados con una variable
	Informa el valor de la variable
	Modifica (incrementa o decrementa) la variable en una cantidad determinada (Si se tiene más de una variable, utilice el menú desplegable para seleccionar el nombre de la variable)
	Fija la variable a un valor específico.
	Muestra el monitor de la variable en el escenario
	Esconde el monitor de la variable para que no aparezca en el escenario
	Permite crear y nombrar una nueva lista. Cuando se genera una lista, aparecen los bloques para esa lista. Se puede escoger si la lista es para todos los Objetos (global) o solo

	Borra los bloques asociados a una lista.
	Reporta (muestra) todos los elementos que tiene la lista.
	Adiciona el elemento especificado al final de la lista (el elemento puede ser un número o una cadena de letras u otros caracteres).
	Borra uno o todos los elementos de una lista. Se puede escoger del menú desplegable o usar un número para indicar qué elemento borrar. Si escoge "último" borrará el último elemento de la lista. Si escoge "todos" borrará todo lo que contiene la lista. Borrar, decrementa la longitud de la lista.
	Inserta un elemento en un lugar específico de la lista. Se puede escoger del menú desplegable o usar un número para indicar dónde insertar el elemento dentro de la lista. Si escoge "último" adiciona el elemento al final de la lista. Si se escoge "cualquiera" lo inserta aleatoriamente en la lista. La longitud de la lista se incrementa en 1.
	Remplaza un elemento de la lista con un valor específico. Se puede escoger del menú desplegable o usar un número para especificar el elemento que va a reemplazar. Si escoge "último", remplaza el último elemento de la lista. Si escoge "cualquiera" remplaza aleatoriamente un elemento de la lista. La longitud de la lista no se modifica.
	Reporta el elemento en una ubicación específica dentro de la lista. Usted puede especificar cuál elemento, eligiendo del menú desplegable o escribiendo un número.
	Reporta cuántos elementos hay en la lista.
	Informa verdadero si la lista contiene el elemento especificado. El item debe coincidir perfectamente para reportarse como verdadero.

ANEXO 3: INSTALACIÓN DE *ENCHANTING*

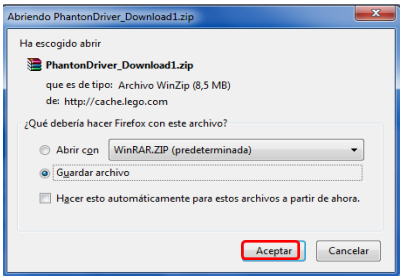
Para poder trabajar con *Enchanting* habrá que tener instalados previamente los programas *Lego Mindstorms NXT*, *Java JDK* y *LeJOS NXJ*. A continuación se explican los pasos a seguir:

Paso 1: Instalación de *Lego Mindstorms NXT*

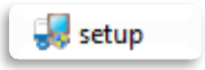
Accedemos al siguiente enlace:

http://mindstorms.lego.com/en-us/support/files/Driver.aspx

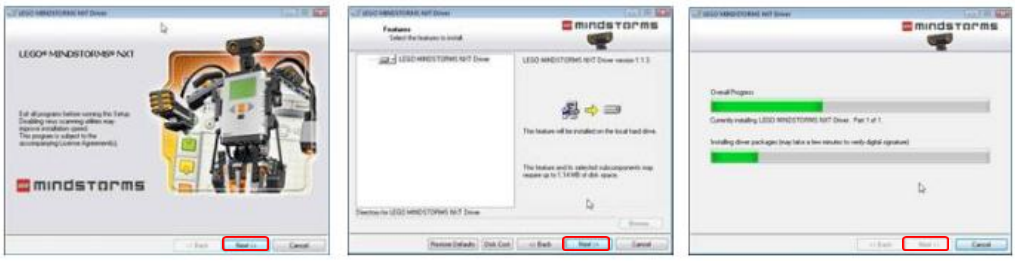
Pulsamos el botón en ***Downloads*** y elegimos la versión compatible con nuestro ordenador (***PC*** o ***MAC***) para iniciar la descarga del archivo:



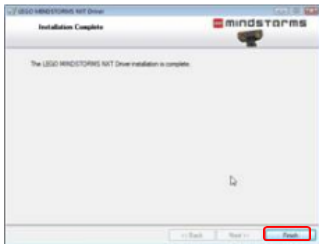
Al finalizar de descargar el ZIP debemos descomprimirlo y acceder al archivo ***setup.exe***; para iniciar el proceso de instalación:



En las tres sucesivas ventanas que irán apareciendo, deberemos pulsar el botón ***Next***:



Al acabar la instalación pulsamos el botón ***Finish***.



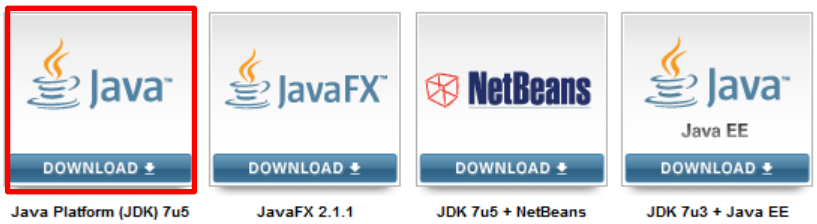
Paso 2: Instalación de *Java JDK*:

Para poder compatibilizar todos los programas descritos, es necesario instalar una versión de Java superior a 1.5. En este caso se describe la instalación de Java JDK 1.7.

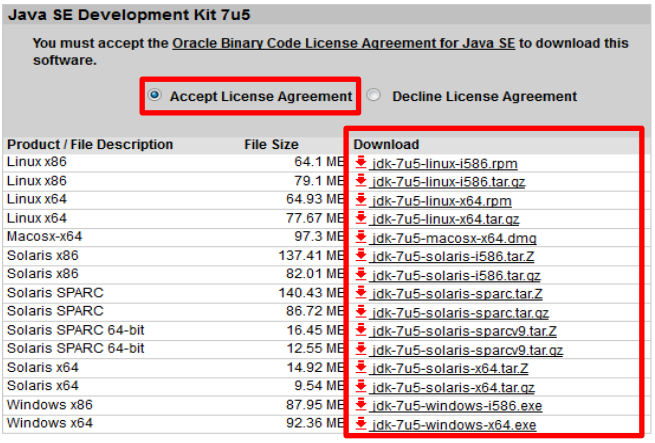
En primer lugar accedemos al siguiente enlace:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

A continuación pulsamos el botón de descarga de *Java JDK 7*:



En la siguiente ventana, debemos aceptar el acuerdo de licencia y elegir el archivo compatible con el sistema operativo que tengamos instalado:



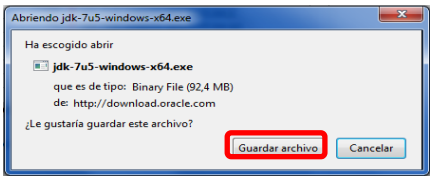
Java SE Development Kit 7u5

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

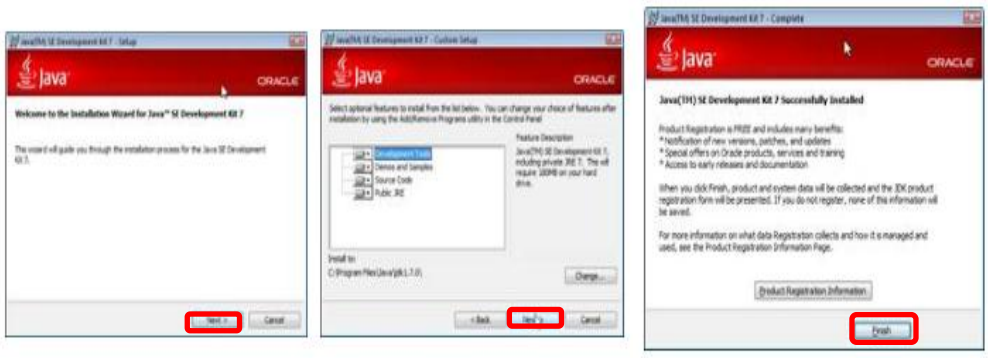
Accept License Agreement | Decline License Agreement

Product / File Description	File Size	Download
Linux x86	64.1 MB	jdk-7u5-linux-i586.rpm
Linux x86	79.1 MB	jdk-7u5-linux-i586.tar.gz
Linux x64	64.93 MB	jdk-7u5-linux-x64.rpm
Linux x64	77.67 MB	jdk-7u5-linux-x64.tar.gz
Macosx-x64	97.3 MB	jdk-7u5-macosx-x64.dmg
Solaris x86	137.41 MB	jdk-7u5-solaris-i586.tar.Z
Solaris x86	82.01 MB	jdk-7u5-solaris-i586.tar.gz
Solaris SPARC	140.43 MB	jdk-7u5-solaris-sparc.tar.Z
Solaris SPARC	86.72 MB	jdk-7u5-solaris-sparc.tar.gz
Solaris SPARC 64-bit	16.45 MB	jdk-7u5-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	12.55 MB	jdk-7u5-solaris-sparcv9.tar.gz
Solaris x64	14.92 MB	jdk-7u5-solaris-x64.tar.Z
Solaris x64	9.54 MB	jdk-7u5-solaris-x64.tar.gz
Windows x86	87.95 MB	jdk-7u5-windows-i586.exe
Windows x64	92.36 MB	jdk-7u5-windows-x64.exe

En ese momento debemos pulsar el botón ***Guardar Archivo*** para descargar el archivo ejecutable a nuestro ordenador:

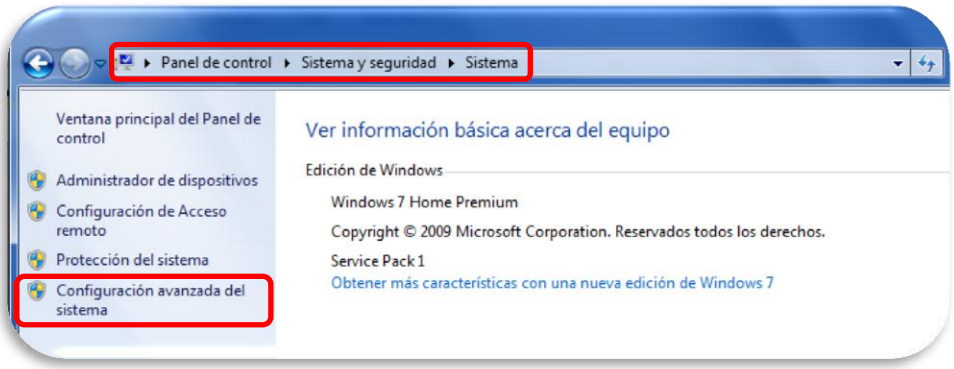


Tras acceder al ejecutable, el proceso de instalación se inicia. Debemos pulsar dos veces la opción ***Next*** en las ventanas que aparezcan y la opción ***Finish*** para finalizar la instalación:

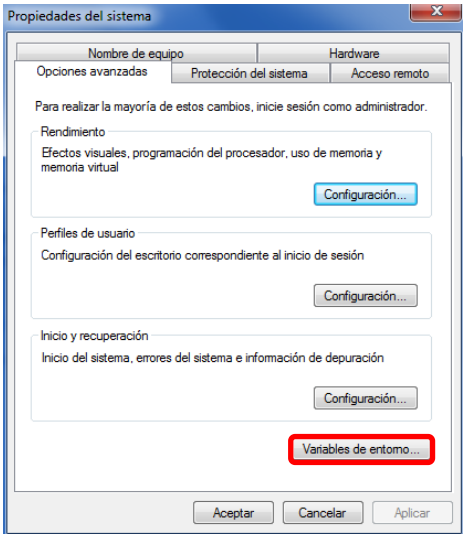


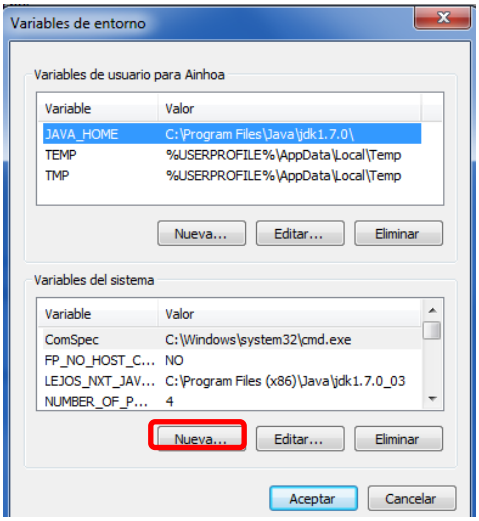
Paso 3: Configuración de las variables del sistema

Para modificar variables de entorno en el ordenador hay que acceder a ***Panel de Control / Sistema y Seguridad / Seguridad / Configuración avanzada del sistema*** (en Windows 7)

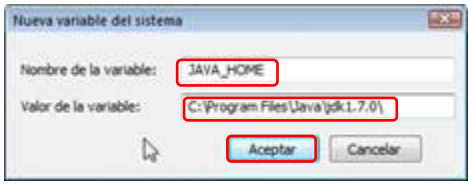


En la nueva ventana pulsamos el botón ***Variables de entorno*** y en la nueva ventana que se abre, creamos una nueva variable para el entorno Java pulsando el botón ***Nueva…***

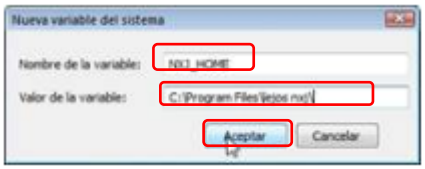




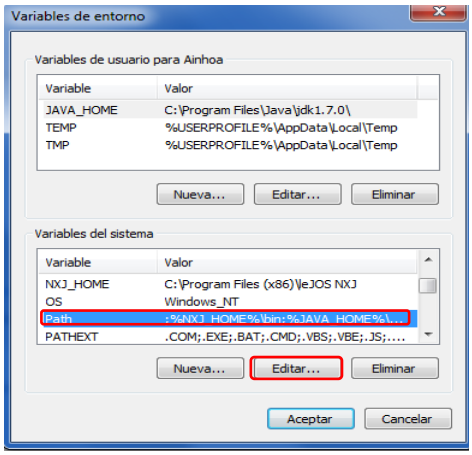
En la nueva ventana que aparece debemos escribir el texto "***JAVA_HOME***" en la primera casilla y en la segunda la ruta al entorno de desarrollo, "***C:\Program Files\Java\jdk1.7.0***" y pulsar el botón ***Aceptar***.

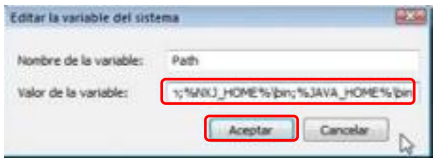


A continuación realizamos la misma operación, creando otra nueva variable, a la que asignamos el nombre "***NXJ_HOME***" y el valor "***C:\Program Files\lejos nxj***"



Para finalizar, es necesario modificar la variable del sistema denominada *Path*. Para ello seleccionamos la variable ***Path*** en la ventana de Variables de entorno y pulsamos el botón ***Editar***. En la segunda casilla de la nueva ventana que aparece, debemos escribir el texto "***;%NXJ_HOME%\bin;%JAVA_HOME%\bin***" y pulsar el botón **Aceptar**.



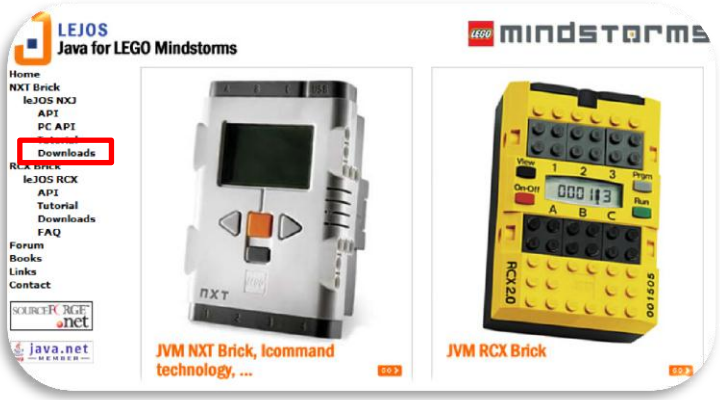


Paso 4: Instalación de LeJOS NXJ

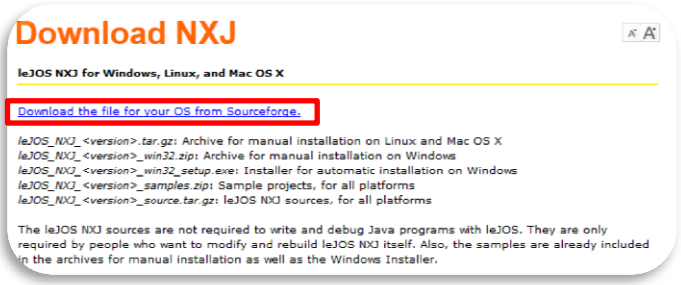
LeJOS NXJ es un entorno basado en Java que nos permite comunicarnos con el cerebro de nuestro robot LEGO, al que se suele llamar *ladrillo NXT*. Existen dos versiones, una para robots NXT y otra para robots RCX, en este caso empleamos la versión para NXT.

Para descargar la aplicación accedemos al siguiente enlace y pulsar la opción ***Downloads*** de **lejos NXJ**.

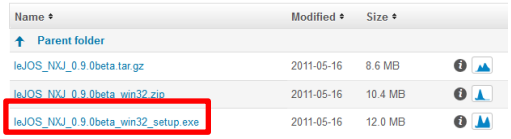
http://lejos.sourceforge.net/index.php



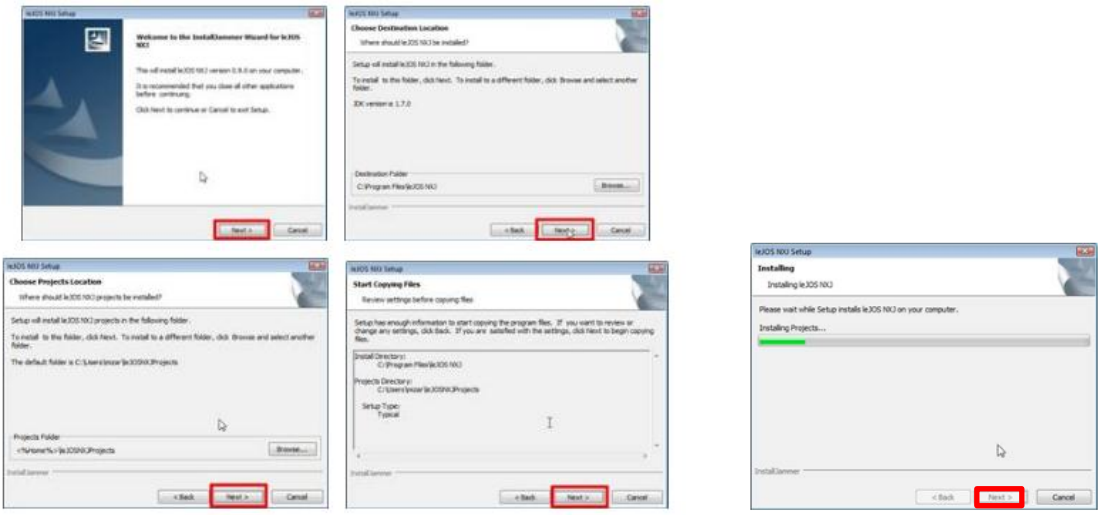
En la nueva ventana clicamos sobre la opción ***Download the file for your OS from Sourceforge***.



En la siguiente ventana seleccionamos el archivo ***0.9.0 beta*** y en la siguiente ventana, escogemos el archivo ***leJOS_NXJ_0.9.0_win32_Setup.exe*** para descargarlo.

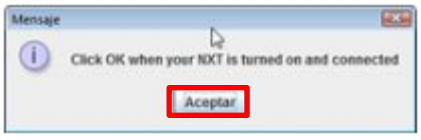


Guardaremos el archivo y lo ejecutaremos para iniciar el proceso de instalación. En las ventanas de este proceso, deberemos pulsar 5 veces el botón ***Next***, para instalar la aplicación. Es importante no modificar la ruta del archivo, para que la variable creada en el paso anterior funcione correctamente.

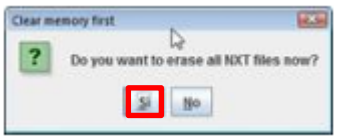


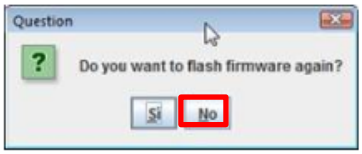
La última ventana del programa no se cierra hasta que actualicemos el software de NXT. Antes de seguir debemos asegurarnos de tener conectado al ordenador el ladrillo NXT, que deberá estar encendido (comprobar que las pilas están bien cargadas).

A continuación debemos pulsar la opción ***Start program*** en la nueva ventana de Java que aparece y pulsar en ***Aceptar*** confirmándole al programa que el ladrillo está conectado y encendido.



Después pulsamos sobre el botón **SI** para confirmar el borrado de los archivos del ladrillo. En este momento se inicia una transferencia de archivos al ladrillo. Al finalizar la transferencia pulsamos sobre el botón **NO** para confirmar que no deseamos volver a descargar los archivos al ladrillo.





ANEXO 4: PROGRAMACIÓN CON *SCRATCH* DEL *RETO 4: VIAJE DE IDA Y VUELTA POR RECORRIDO DEFINIDO POR PARADAS*

En este apartado se expone un ejemplo para programar el Reto 4 (*Viaje de ida y vuelta por recorrido definido por paradas)* con *Scratch*:

Para este reto, se diseñan 5 objetos que serán las paradas con las siguientes coordenadas (X,Y): (-200,-30), (-100,-30), (0,-30), (100,-30) y (200,-30).

Para cada parada se diseñan dos disfraces, uno para usarlo cuando el autobús no está en ella y otro para usarlo cuando el autobús llegue.

A continuación se crea una lista con las coordenadas en X de las 5 posiciones. Para ello se emplea la función *Nueva Lista* del bloque de *Variables* y la función *añade (1) a (2).* En este caso, (1) serán las X y (2) el nombre de la lista creada. En el ejemplo la lista se llama *Pos_Paradas*.

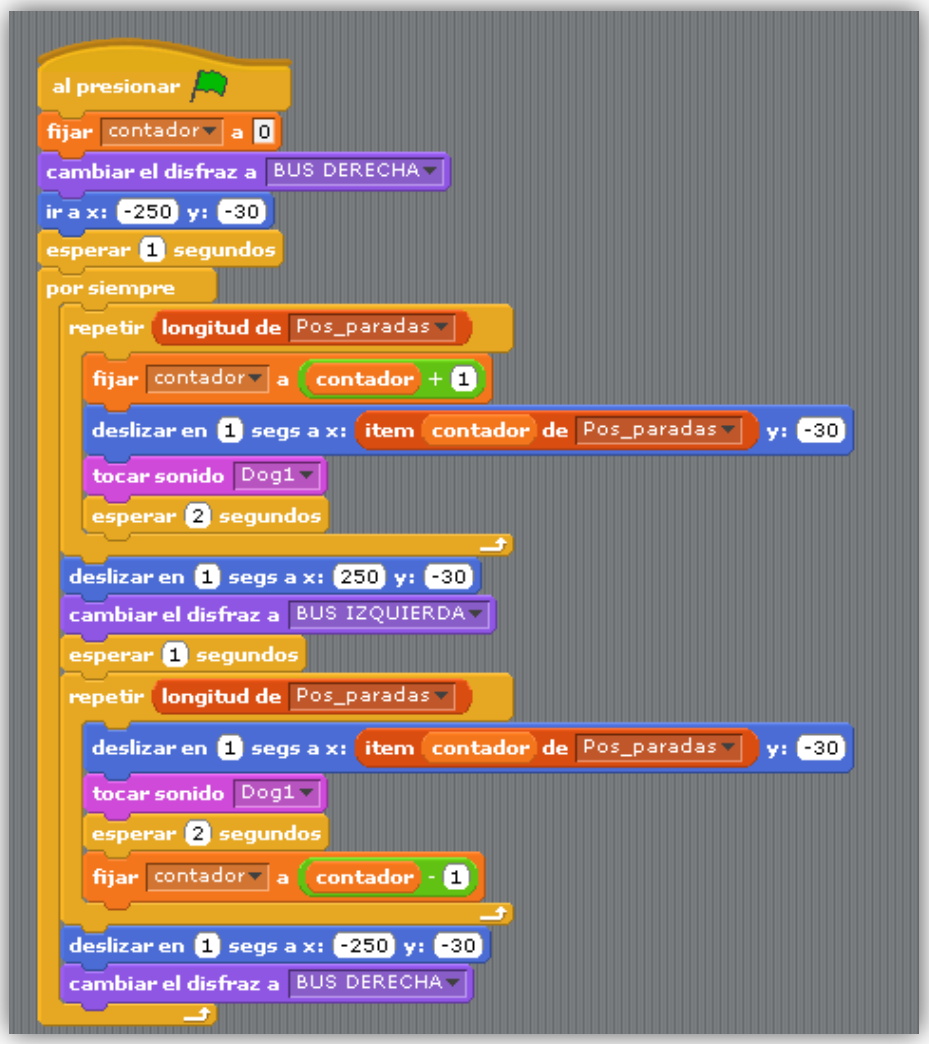
Crearemos también una variable numérica, que nos sirva para conocer en qué posición está el autobús. Para ello utilizamos la función Nueva Variable del bloque de Variables. La variable en este ejemplo se llama *contador*.

Después se diseña el objeto que se irá desplazando, en este caso se han diseñado dos disfraces, uno para la ida y otro para la vuelta. En el inicio del programa, situaremos este objeto en la posición (-250, -30).

Ahora que tenemos los elementos, comenzamos la secuencia de instrucciones para el desplazamiento del objeto:

- La primera instrucción es fijar el contador a cero, ya que de momento no se ha desplazado.
- A continuación le asignamos al objeto su disfraz de Ida.
- Después inicializamos la posición del objeto a (-250, -30).
- Esperamos 1 segundo antes de iniciar el recorrido.
- Empezamos el bucle infinito:
- Usamos un bucle de repetición, hasta que se realice el recorrido de ida por las cinco paradas.
  El bucle se repite 5 veces, una por cada parada. Para saber en qué parada está, cada vez que se entra en el bucle, sumamos uno al contador.
  Dentro del bucle, usamos la función de deslizamiento en 1 segundo a la posición de la parada que toca. Para saber a qué parada se tiene que desplazar, usamos el contador para acceder a los ítems de la lista. La 1ª vez, el contador estará a 1 y el autobús se deslizará hasta la posición definida en ítem 1 de la lista Pos_Paradas.
  Tras el desplazamiento, se alerta mediante un sonido (Dog) de que el autobús ha llegado a la parada.
  Por último, el objeto espera un par de segundos en cada parada, antes de realizar el bucle para la siguiente posición.
- Antes de empezar el bucle correspondiente al viaje de vuelta, desplazamos el objeto hasta la posición (250, 30), cambiamos su disfraz por el correspondiente a la vuelta y esperamos 1 segundo antes de iniciar la vuelta.
- Por último repetimos el bucle anterior, pero en este caso cada vez que se realiza el bucle, se resta uno al contador.
- Para acabar, deslizamos el objeto hasta la posición (-250,-30) y cambiamos el disfraz para un nuevo recorrido hacia la derecha.

El Bloque de instrucciones se puede ver en la siguiente imagen:



En este reto, debemos conseguir que cada parada cambie de disfraz cuando llegue el autobús. Para ello a cada parada le corresponden las siguientes instrucciones:

- Cada parada emplea un bucle infinito y una estructura condicional. Se recurre a la función *tocando_* del bloque *Sensores*. En este caso la instrucción es que si la parada está tocando al objeto móvil (Sprite 1) cambie al disfraz que alerta de que el autobús está en este punto. En caso contrario debe ponerse el disfraz que indica esta circunstancia. En el ejemplo los disfraces se denominan HAY BUS y NO HAY BUS respectivamente.

A continuación se muestra la secuencia de instrucciones que se repite para todas las paradas:

ANEXO 5: PROGRAMACIÓN CON *ENCHANTING* DEL *RETO 6: ROBOT SIGUE LÍNEAS*

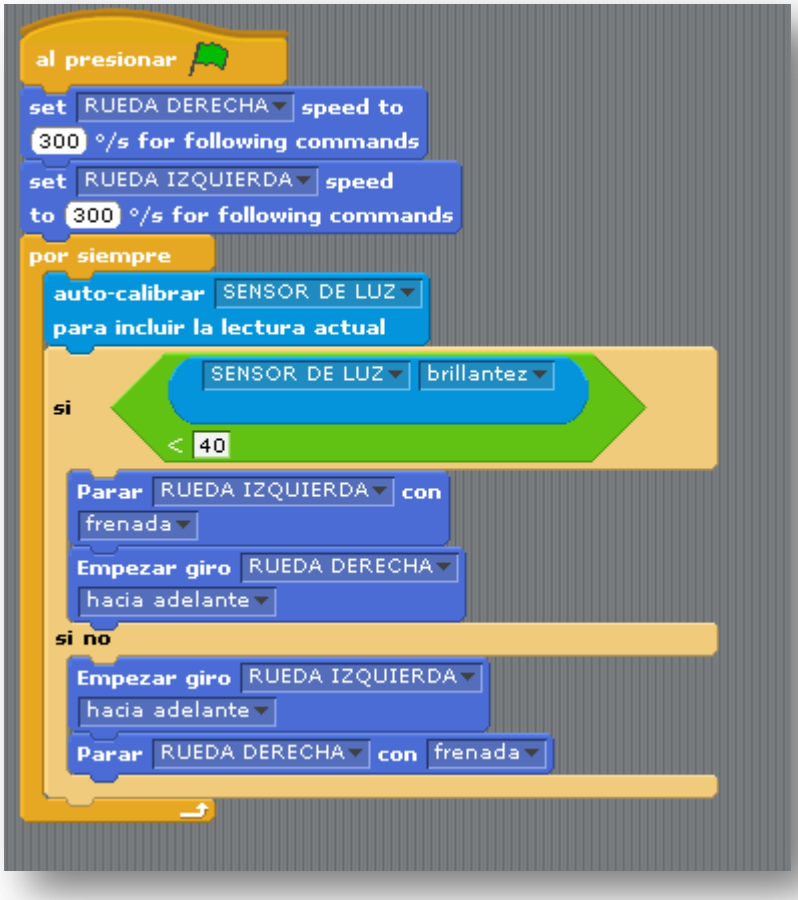
En este apartado se expone un ejemplo para programar el Reto 6 (*Robot Sigue Líneas)* con *Enchanting*. En este caso, el robot sigue una línea de color negro dibujada en un espacio de color blanco. Para llevar a cabo este reto, se emplea un sensor de luz y dos motores NXT, además del ladrillo.

Con la primera Instrucción, activamos nuestro programa.

Las dos siguientes instrucciones, definen la velocidad de los motores NXT (RUEDA IZQUIERDA y RUEDA DERECHA) en 300° / segundo.

Dentro del bucle infinito, primero autocalibramos el sensor de luz, para actualizar los valores.

A continuación, si el sensor de luz detecta un valor < 40 (línea negra) gira la el sensor derecho y el izquierdo se detiene. En caso contrario (espacio blanco), gira el sensor izquierdo y el derecho se detiene.



BIBLIOGRAFÍA

[1] Datos económicos:
http://es.wikipedia.org/wiki/Anexo:Pa%C3%ADses_por_PIB_%28nominal%29_per_c%C3%A1pita

[2] Instalación y descarga de *Scratch*:

http://scratch.mit.edu/

[3] Mini-guía de sensores *LegoWedo* compatibles con *Scratch*:

http://scratched.media.mit.edu/sites/default/files/WeDoMiniguide.pdf

[4] Kinect:

http://scratched.media.mit.edu/discussions/teaching-scratch/scratch-kinect

[5] Sensores Virtuales:

http://softwareybarralibre.org/?q=content/sensores-virtuales-para-scratch-enchanting-y-s4a

[6] *Lego Mindstorms NXT*

http://lrobotikas.net/eu/como-empezar/59-lego-mindstorms/86-hm04

http://www.ehu.es/ikastorratza/6_alea/lego.pdf

[7] Instalación de *Enchanting*:

http://recursostic.educacion.es/observatorio/web/es/software/software-educativo/1018-monograficodesarrollos-de-scratch-para-robotica-enchanting-y-s4a?start=2

[8] PBL

http://www.vcu.edu/cte/resources/nfrg/11_07_problem_based_learning.htm